ENTWICKLUNG EINER „RICH INTERNET APPLICATION“ ZUR INTERAKTIVEN VISUALISIERUNG STATISTISCHER MASSENDATEN

Diplomarbeit

im Fachbereich Photoingenieurwesen und Medientechnik
an der Fachhochschule Köln,
Fachrichtung Medientechnik

In Zusammenarbeit mit dem
Atelier für Mediengestaltung
Doering, Kern, Günther GbR

Autorin
Janina Alexandra Maria Werner
Mat.-Nr. 11048027

Referent: Prof. Dr. rer. nat. Stefan Grünvogel
Korreferent: Dipl. Photoing. Frank Doering

Köln, im August 2009

DEVELOPMENT OF A "RICH INTERNET APPLICATION" FOR INTERACTIVE VISUALIZATION OF STATISTICAL MASS DATA

Thesis at the Department
of Imaging Sciences and Media Technology
University of Applied Sciences Cologne

With the collaboration of
Atelier für Mediengestaltung
Doering, Kern, Günther GbR

Author
Janina Alexandra Maria Werner
Mat.-Nr. 11048027

First Reviewer: Prof. Dr. rer. nat. Stefan Grünvogel
Second Reviewer: Dipl. Photoing. Frank Doering

Cologne, August 2009

Titel: Entwicklung einer „Rich Internet Application“ zur interaktiven Visualisierung statistischer Massendaten

Autorin: Janina Alexandra Maria Werner

Referenten: Prof. Dr. rer. Nat. Stefan Grünvogel / Dipl. Ing. Phot. Frank Doering

Zusammenfassung:

Das permanente Angebot und die Nachfrage an Informationen und Daten jeglicher Art wachsen zunehmend. Das Ergebnis einer meist verschachtelten Suche nach bestimmten Zahlen ist jedoch oftmals eine unübersichtliche, tabellarische Aufstellung derer. Zudem ist dagegen die Aufnahme grafischer Informationen erheblich höher und effektiver. Als Ergebnis der Diplomarbeit soll im ersten, theoretisch-wissenschaftlichen Teil eine technische Konzeption für ein intelligentes Visualisierungs-System erarbeitet werden. Im zweiten, praktischen Teil soll anhand der untersuchten Methoden und des entwickelten Konzepts eine Anwendung kreiert werden, welche ausgewählte Methoden zur interaktiven Visualisierung statistischer Daten nutzt.

Stichwörter: Rich Internet Application, Daten-Visualisierung, Flex 3, Actionscript 3, Internet

Sperrvermerk: Die Einsicht in die vorgelegte Arbeit ist bis zum 28.08.2011 gesperrt

Datum: 28.08.2009

Title:	Development of a “Rich Internet Application” for interactive visualization of statistical mass data
Author:	Janina Alexandra Maria Werner
Advisors:	Prof. Dr. rer. nat. Stefan Grünvogel / Dipl. Ing. Phot. Frank Doering

Abstract:

The permanent supply and demand of all types of information and data are growing increasingly. But often the result of a mostly nested searching for specific data is a confusing array in table form. Furthermore in contrast, the realisation of graphical information is much more effective. As conclusion of this thesis there is to be developed a technical conception for an intelligent visualisation system, in the first, the theoretical-scientific part; on the other hand, there is to be created an application, which uses selected methods for interactive visualisation of statistical data, in the second, the applied part.

Key words:	Rich Internet Application, Data-Visualization, Flex 3, Actionscript 3, Internet
Remark of closure:	The thesis is closed until 2011.08.28
Date:	28.08.2009

Inhaltsverzeichnis

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.

Mein besonderer Dank gilt der Firma Atelier für Mediengestaltung. Insbesondere danke ich den Geschäftsführern Frank Doering, Frank Günther und Tobias D. Kern für die interessante Aufgabenstellung und die großartige Unterstützung während des gesamten Diplom-praktikums.
Meinem Projektteam, Matthias Müller und Simon Widjaja, möchte ich ebenfalls für die tolle Zusammenarbeit danken.

Weiterhin möchte ich mich bei Prof. Dr. rer. nat. Stefan Grünvogel bedanken, der mir von Seiten der Fachhochschule Köln bei Fragen jederzeit zur Verfügung stand.

Zu guter Letzt danke ich herzlich meiner Familie, besonders meinen Eltern Marion und Alexander, ohne deren finanzielle Unterstützung mein Studium nicht möglich gewesen wäre und meinem Partner Mike, der mir mit Motivation und Verständnis stets zur Seite stand.

Danke!

1 Einleitung und Motivation

Das permanente Angebot und die Nachfrage an Informationen und Daten jeglicher Art wachsen in der heutigen Zeit zunehmend. Viele Statistikportale stehen im Internet zur Verfügung und bieten den Zugriff auf ihre Datenbanken kostenfrei oder gegen eine Lizenz an.

Oft jedoch ist das Ergebnis einer meist verschachtelten Suche nach bestimmten Zahlen eine unübersichtliche, tabellarische Aufstellung derer. Alleine das Statistische Bundesamt Deutschland, auf das sich diese Arbeit beispielhaft bezieht, bietet über den zentralen Datenzugang mit dem Namen *GENESIS-Online*[1] (Gemeinsames neues statistisches Informationssystem) rund 1000 Tabellen zur Recherche und zum Download an. Das menschliche Auge kann jedoch nur in einem begrenzten Maße geschriebene Informationen, in diesem Fall Zahlen, wahrnehmen und verarbeiten. Dahingegen ist die Aufnahme grafischer Informationen erheblich höher und effektiver. Da es nur schwer möglich ist, statistische Zahlen in Tabellenform miteinander vergleichend ins Verhältnis zu setzen, müssen solche Fragestellungen manuell bearbeitet werden. Hierbei können wertvolle Informationen und vor allem Zusammenhänge zwischen den unterschiedlichen Daten unbeachtet bleiben.

Es wird deutlich, dass gerade dem Betrachter, welcher keinen täglichen Umgang mit derartigen Statistiken hat, der Zugang zu diesen interessanten und aufschlussreichen Daten weitgehend verwehrt bleibt. So steht die Frage im Raum, wie eben diese Daten einfach und anschaulich visualisiert werden können, so dass daraus Informationen und persönliche Erkenntnisse für jedermann zu generieren sind.

[1] https://www-genesis.destatis.de/genesis/online

Ziel dieser Arbeit ist die Konzeption und Entwicklung einer *Rich Internet Application* (kurz RIA) zur interaktiven Visualisierung statistischer Massendaten. Dazu gehören sowohl eine umfassende Recherche, die als Querschnitt und Bewertung aktueller Visualisierungsansätze in Praxis und Forschung zu verstehen ist, als auch die Entwicklung eines Visualisierungssystems, in das einzelne interessante Module zu den im theoretischen Teil der Arbeit vorgestellten Methoden integriert werden. Alle theoretischen Grundlagen und erarbeiteten Zusammenhänge werden in dieser Arbeit vorgestellt und diskutiert. Abschließend bietet die Zusammenfassung einen kritischen Ausblick über mögliche weitere Funktionalitäten und Erweiterungen der Anwendung.

2 Grundlagen der Visualisierung

Dieses einführende Kapitel ist Spiegelbild einer ausführlichen Recherche und soll den hohen Stellenwert einer grafischen Visualisierung von Daten in der heutigen Zeit verdeutlichen. Der Begriff *Daten* steht dabei nicht nur für Zahlen und Werte, sondern kann, wie in den folgenden Ausführungen erläutert, auf unterschiedliche Weise verstanden werden, da Visualisierungen in den verschiedensten Bereichen Verwendung finden [vgl. Kap. 2.4].

2.1 Zeichen, Daten, Information, Wissen

Um die Bedeutung grafischer Visualisierungen von Zahlen und deren Zusammenhänge klarer herausstellen zu können, folgt zunächst die Klärung der vier in diesem Kontext immer wiederkehrenden Begriffe *Zeichen*, *Daten*, *Information* und *Wissen* und deren Abgrenzung voneinander.

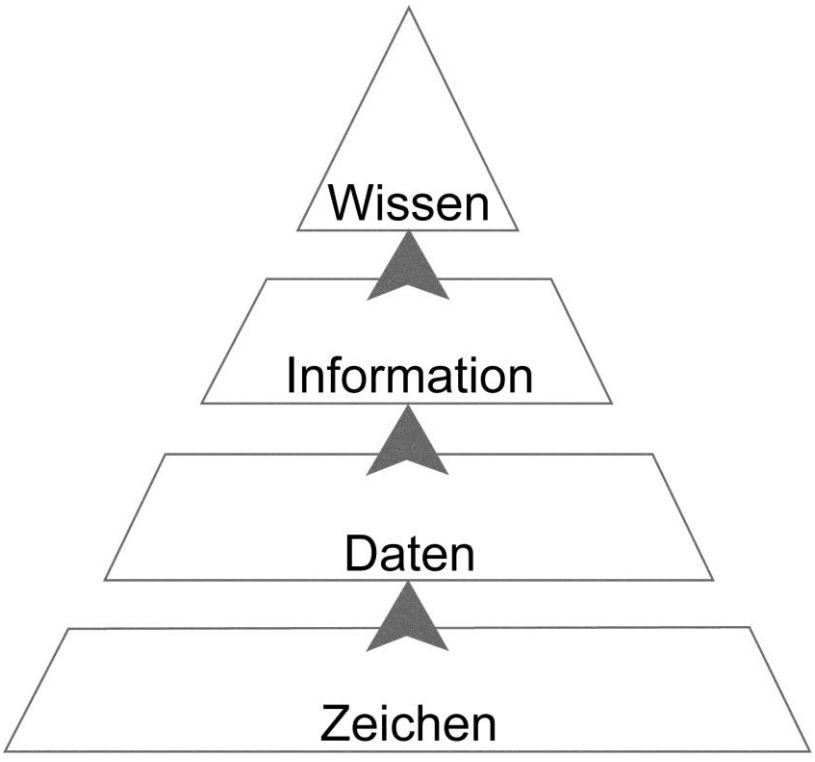


Abb. 1 - Begriffshierarchie: Zeichen, Daten, Information und Wissen

„Wissen […] entsteht als Folge eines komplizierten Prozesses der Verarbeitung, Filterung und Bewertung von Informationen.“ [VSS07].

> „Zeichen, Informatik: englisch Character, Element aus einer zur Darstellung von Information vereinbarten endlichen Menge (Zeichenvorrat).“ [ML009a]

Zu Beginn dieses Prozesses steht das Zeichen:
„Als Zeichen gelten Buchstaben, Ziffern und Sonderzeichen“, „… die durch bestimmte Ordnungsregeln, wie zum Beispiel einem Code oder einer Syntax, in Beziehung zueinander gesetzt werden.“ [VSS07].

> „Daten [zu Datum], Informatik: zur Darstellung von Informationen dienende Zeichenfolgen (digitale Daten) oder kontinuierliche Funktionen (analoge Daten), die auf Datenverarbeitungsanlagen gespeichert, verarbeitet und/oder erzeugt werden können.[…]“ [ML009b]

Zeichenfolgen werden als Daten bezeichnet.
Diese Zeichenketten sind Daten, „die […] noch nicht interpretierbar und somit potenziell verwertbar sind. Erst wenn Daten mit Bedeutung versehen und ein subjektiver Bezug hergestellt wird, werden Informationen erzeugt, …“ [VSS07].

> „Information [lateinisch] die, Informatik: „in Form gebrachtes“ Wissen wie Mitteilungen, Nachrichten, die Gegenstand von Speicherung, Verarbeitung und Übertragung sind, meist dargestellt als eine Folge von Zeichen aus einem bestimmten Zeichenvorrat; in der Informationstheorie ein technisches Maß, das den Zeichen einer Nachricht zugeordnet wird. […]“ [ML009c]

Wissen wiederum ist das Resultat der Verarbeitung von Informationen sowie der Umgang mit Daten und Informationen. Diese zu strukturieren und zu verstehen, setzt individuelles und kollektives Wissen voraus.

2.2 Die Geschichte der Datenvisualisierung

Der Begriff *Visualisierung* oder *Veranschaulichung* stammt aus dem Lateinischen („videre“ = „sehen“) und umfasst ganz allgemein das Sichtbarmachen von abstrakten Daten und Zusammenhängen. Die grafische oder visuell erfassbare Form soll dabei das Verständnis über die zugrunde liegenden Daten erleichtern und innere, meist verborgene Zusammenhänge aufzeigen. Nur so ist es dem Betrachter möglich, Schlüsse zu ziehen, zu verstehen und zu bewerten.

Visualisierungen sind dabei nicht etwa ein Produkt jüngster Techniken, sondern fanden schon weit vor dem 17. Jahrhundert in verschiedenen Bereichen Verwendung. Karten und Diagramme, in Form von Balken, Linien, Kreisen und Zeitleisten, existieren seit mehreren Jahrhunderten. In „Milestones in the history of thematic cartography, statistical graphics, and data visualization“ stellen Michael Friendly und Daniel J. Denis verschiedenste Formen der Visualisierung aus historischer Zeit bis heute zusammen [FD06]: Die Wurzeln heutiger Grafiken lassen sich bis ins 16. Jahrhundert zurückverfolgen.
So wurde beispielsweise das schon von Leonardo da Vinci (Maler, Bildhauer, Architekt, Anatom, Mechaniker, Ingenieur und Naturphilosoph, 1452-1519) genutzte kartesische Koordinatensystem von René Descartes (französischer Philosoph, Mathematiker und Naturwissenschaftler 1596-1650) wieder entdeckt und findet bis heute auf dem Gebiet der Mathematik und analytischen Geometrie Verwendung. Land-, See- und Himmelskarten ermöglichten als kartografische Werke abstrahierte Darstellungen mit geografischem Bezug. Die Entwicklung von Techniken und Instrumenten für präzise Messungen, durch beispielsweise Tycho Brahe (dänischer Adliger und einer der bedeutendsten Astronomen, 1546-1601), trieben den Fortschritt in der Datenvisualisierung voran.

Bereits im 17. Jahrhundert kannte man die analytische Geometrie (Koordinatensystem), die Fehlertheorie von Mess- und Schätzverfahren sowie die Anfänge der Wahrscheinlichkeitstheorie und der

demografischen Statistik (Bevölkerungsstatistik). Das Monumentalwerk „Encyclopédie, ou Dictionnaire raisonné des sciences, des arts et des métiers“ von Jean Baptiste le Rond d’Alembert und Denis Diderot, welches zwischen 1751 und 1780 entstand, beinhaltet in 35 Bänden 299 Informationsgrafiken über Handwerk und Handel [DD1780]. Für die Darstellung immer abstrakter werdender Sachverhalte kamen nach und nach die heute noch vielseitig verwendeten Diagrammformen (Linien- und Balken- sowie Kreisdiagramme) hinzu, so auch vielfach im „Commercial and Political Atlas and Statistical Breviary“ von William Playfair (schottischer Ingenieur und Volkswirt, 1759-1823) verwendet.

Während die erste Hälfte des 19. Jahrhunderts schon für einen beachtlichen Boom an Statistikgrafiken (Balken- und Kreisdiagramme, Histogramme, Linien- und Zeitreihengrafiken, Kontur- bzw. Querschnittplots, Streudiagramme u.v.m.) verantwortlich war, stellte die zweite Hälfte des 19. Jahrhunderts das goldene Zeitalter derer dar. Erstmals wurden für die Organisation numerischer Informationen staatliche Statistikämter ins Leben gerufen.

Neben dem Visualisieren auf Papier begann 1957 die Computerverarbeitung von statistischen Daten. Seit 1975 spricht man von der *High-D-Daten-Visualisierung* (engl. High Dimensional Data Visualization).

2.3 Die Aufgaben der Informationsvisualisierung

Um aus Daten Informationen zu gewinnen, müssen diese in einen Kontext gebracht und evtl. organisiert werden. Diese visuelle Aufbereitung von Daten entspringt dem Forschungsgebiet der Informationsvisualisierung (engl. Information Visualization). Mithilfe grafischer Darstellungsmethoden sollen Auswertung und daraus folgende Erkenntnisgewinnung vereinfacht werden.

In diesem Zusammenhang erwähnt Rolf Dässler vier Merkmale und Aufgaben der Visualisierung [Dae09]:

- Visuelle Darstellungen helfen, komplexe Prozessabläufe und Objektbeziehungen in der Realwelt zu veranschaulichen und ggf. zu vereinfachen.
- Visualisierung vereinfacht den Zugang zu Massendaten, z.B. durch Klassifikation und Datenstrukturierung.
- Visualisierung hilft bei der Analyse und Interpretation von Daten, bei der Sichtbarmachung verborgener Trends sowie bei der Mustererkennung.
- Visualisierung entspricht der Neigung der Menschen und ihrer Kultur, visuelle Informationsprozesse und Repräsentationsformen zu bevorzugen. Darüber hinaus ist aus der Gehirnforschung bekannt, dass sich Visualisierung positiv auf die Gedächtnisleistung und auf die Informationsaufnahme auswirkt.

2.4 Visualisierungsbeispiele im Internet

Neben der Visualisierung von Statistiken in Diagrammen und Infografiken existiert im Internet zudem eine große Anzahl weiterer moderner, teilweise experimenteller, Visualisierungsformen:

digg labs[2] präsentieren beispielsweise unter dem Namen *bigspy* aktuelle Beiträge durch die digg.com-Community in Form unterschiedlich großer Schlagzeilen, die sich vertikal über den Bildschirm bewegen. Die Größe dieser Schlagzeilen spiegelt die Häufigkeit der Klicks zu dem relevanten Artikel wider.

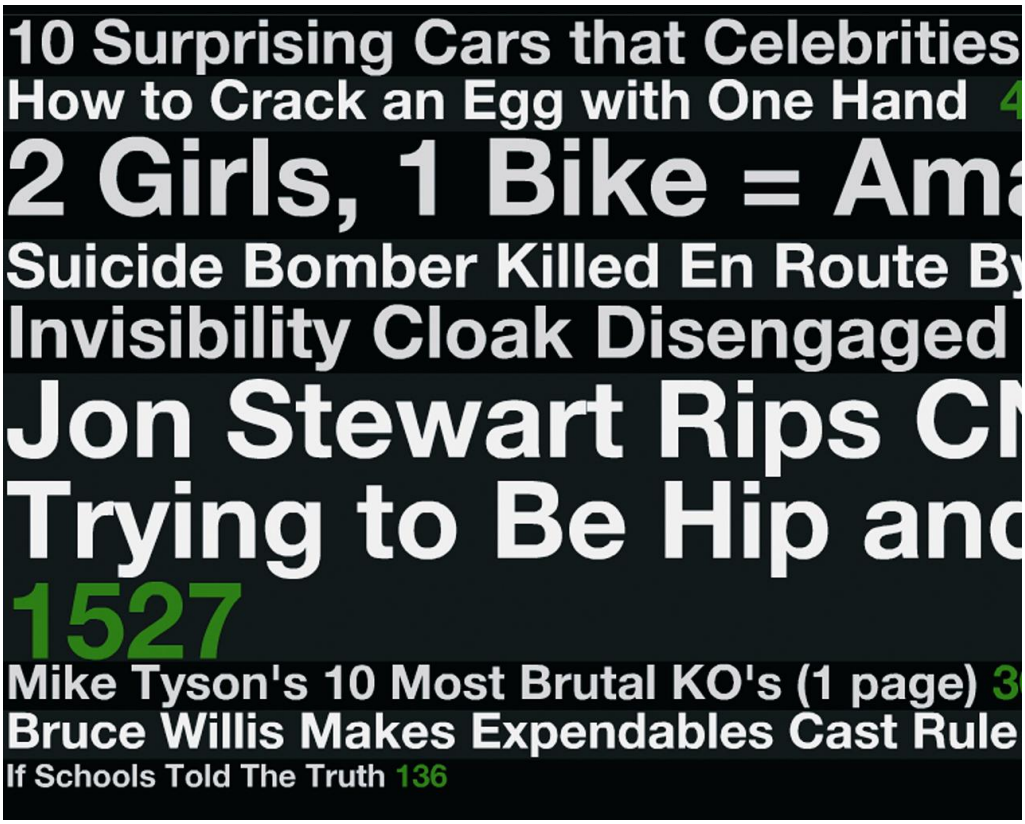


Abb. 2 - digg labs

[2] http://labs.digg.com/bigspy

Ein weiteres Beispiel für die Visualisierung im Internet ist *Gapmider World*[3] von der *Gapminder Foundation*. Es stellt ein zentrales Tool für die Visualisierung von Daten aller Art kombiniert miteinander dar. Anhand zweier frei wählbarer Achsen und der Auswahlmöglichkeit aller Länder werden dem Benutzer Daten im Hinblick auf die kontinentale Zugehörigkeit (Farbe der Kreise) und Bevölkerungsgröße (Größe der Kreise) dargestellt. Zusätzlich ist es möglich, eine zeitliche Veränderung dieser Daten im dreißigjährigen Rückblick bis einschließlich 2007 über einen animierten Zeitstrahl zu visualisieren.

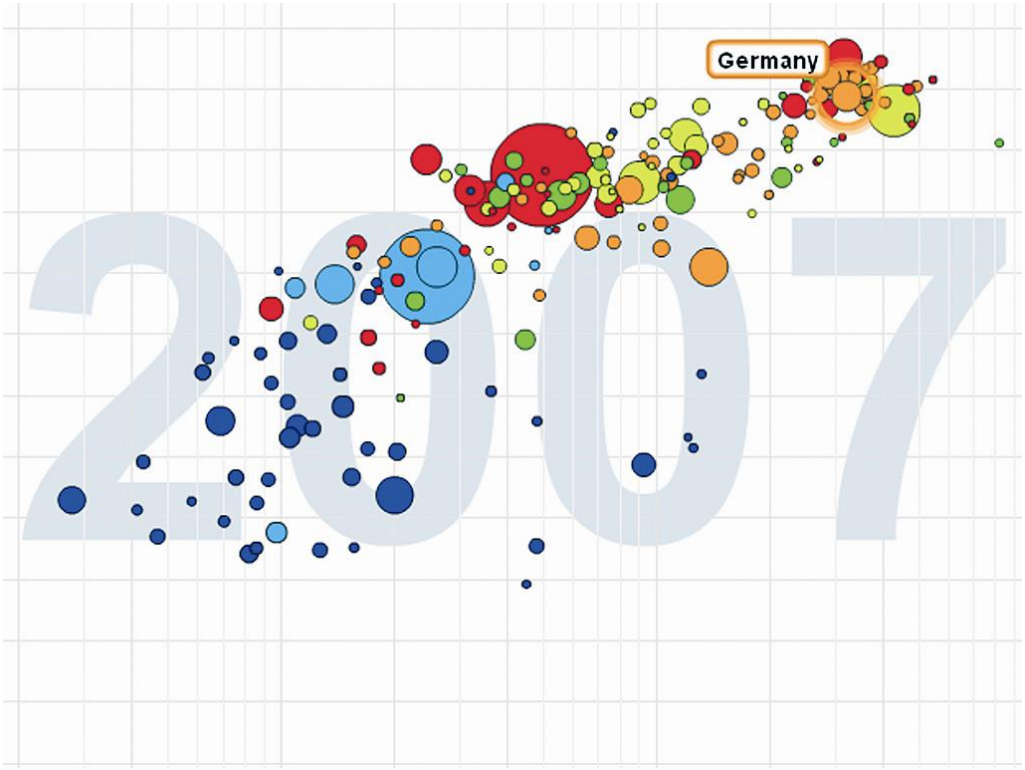


Abb. 3 - Gapminder World

[3] http://graphs.gapminder.org/world

Eine Analyse über den Zeitaufwand verschiedenster Bevölkerungsgruppen für ihre täglichen Aktivitäten bietet die *New York Times* in ihrer interaktiven Visualisierung *How Different Groups Spend Their Day*[4]. Eine Achse steht für die Tageszeit, die andere Achse für den prozentualen zeitlichen Anteil einer Tätigkeit im Hinblick auf einen ganzen Tag. Beschäftigungen, die unterschieden werden, sind: Essen, Arbeiten, Hausarbeit, Reisen, Fernsehen, soziale Kontakte pflegen und Schlafen.

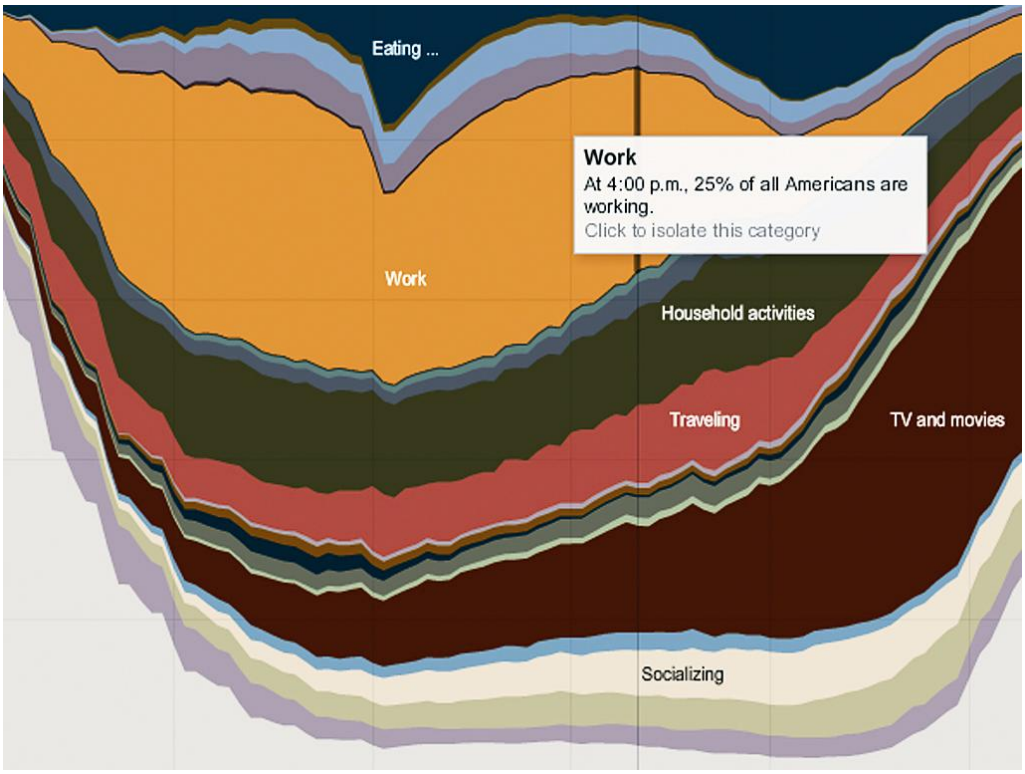


Abb. 4 - How Different Groups Spend Their Day

[4] http://www.nytimes.com/interactive/2009/07/31/business/20080801-metrics-graphic.html

Die Visualisierung von Internetseiten in Form von Graphen ermöglicht *Webpages as Graphs*[5]. Nach Eingabe einer beliebigen URL wird ein Graph geladen, der aus verschieden farbigen Punkten und einer verbindenden dünnen Linie besteht. Die Farben der Punkte stehen für folgende Informationen über die Internetpräsenz:

blue:	für Links (A-Tag)
red:	für Tabellen (TABLE-, TR- und TD-Tags)
green:	für DIV-Tags
violet:	für Bilder (IMG-Tag)
yellow:	für Formulare (FORM-, INPUT-, TEXTAREA-, SELECT und OPTION-Tags)
orange:	für Zeilenumbrüche und Blockinhalte (BR-, P- und BLOCKQUOTE-Tags)
black:	für HTML-Tags
gray:	alle anderen Tags

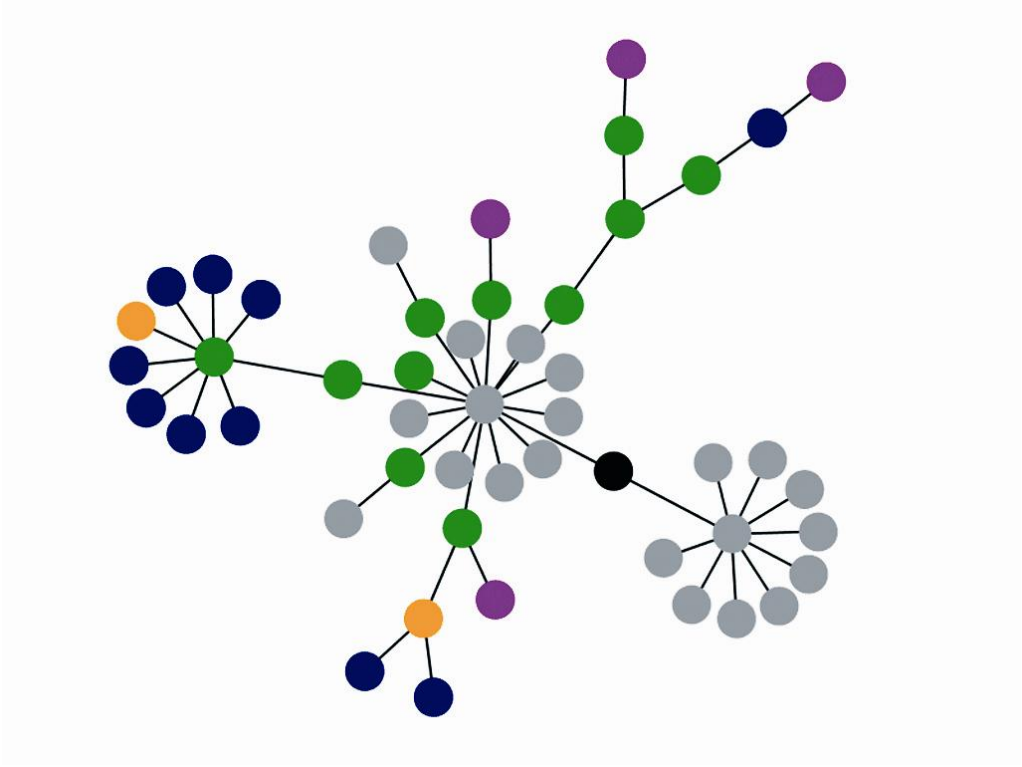


Abb. 5 - Webpages as Graphs

[5] http://www.aharef.info

In enger Zusammenarbeit mit der Hochschule für Gestaltung und Kunst Zürich entwickelten Magnus Rembold und Jürgen Späth unter dem Namen *Munterbund*[6] ein System zur Visualisierung der 19 verschiedenen Kapitel aus dem Buch „Total Interaction“. Dies ist ein Werk, „das die unterschiedlichen Sichtweisen auf die Interaktion als Feld der Mensch-Maschine-Kommunikation aufzeigt.“ [RS04]

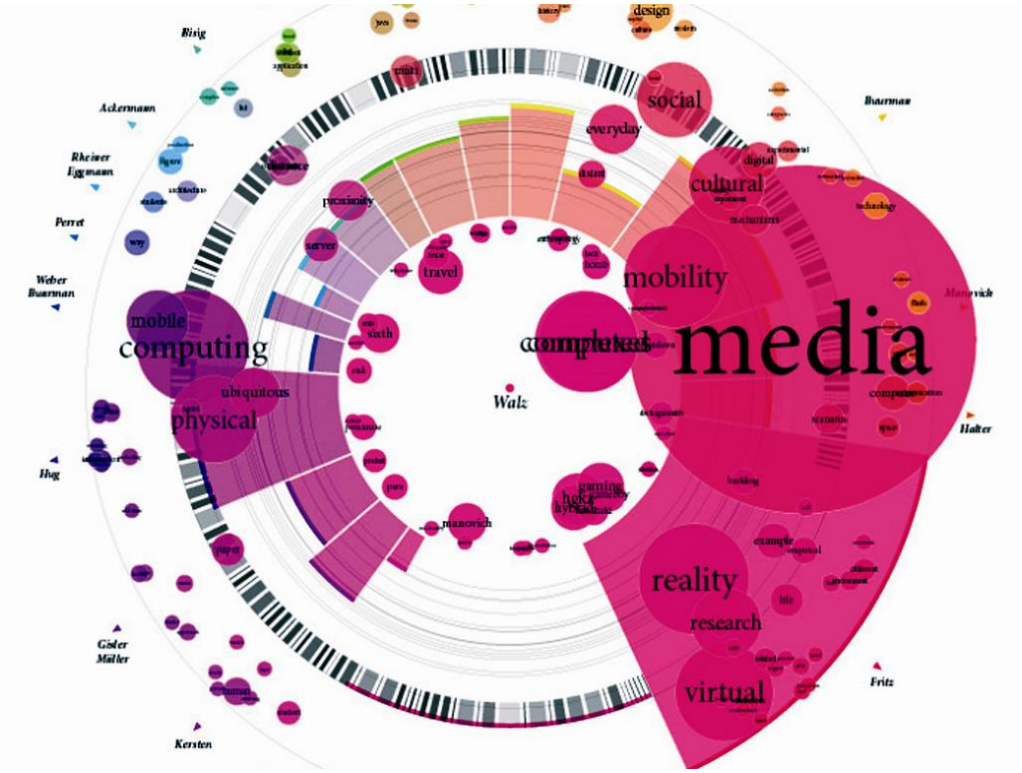


Abb. 6 - Munterbund

[6] http://www.munterbund.de/visualisierung_textaehnlichkeiten/artikel.php

2.5 Statistische Infografiken

Anders als in der Tabellenform werden durch statistische Infografiken nicht nur Zahlen sondern auch deren Zusammenhänge veranschaulicht.

Es werden Balken- und Säulendiagramme, Linien- oder Kurvendiagramme sowie Torten- oder Kreisdiagramme unterschieden. Eine weitere Form der Infografik ist die thematische Karte.

2.5.1 Balken- und Säulendiagramme

Balken- und Säulendiagramme sind die am meisten gebräuchliche Darstellungsform für Mengenvergleiche (bis ca. 15) und unterscheiden sich lediglich in der Ausrichtung: Balken verlaufen horizontal, Säulen dagegen vertikal. Zusätzlich sind zeitlich verlaufende Messreihen darstellbar.

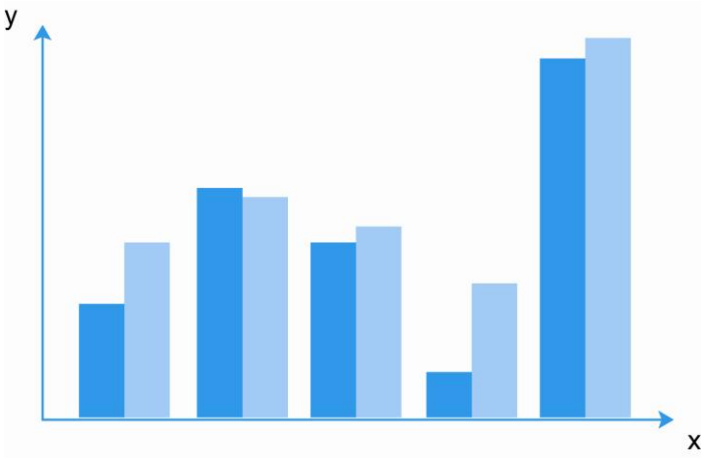


Abb. 7 - Beispiel für ein Säulendiagramm

2.5.2 Linien- oder Kurvendiagramme

Auch unter dem Namen *Fieberkurven* bekannt, zeigt diese Diagrammform die Verbindung einzelner Messpunkte, die zwischen x- und y-Achse des Koordinatensystems abgetragen wurden, in einer Linie.

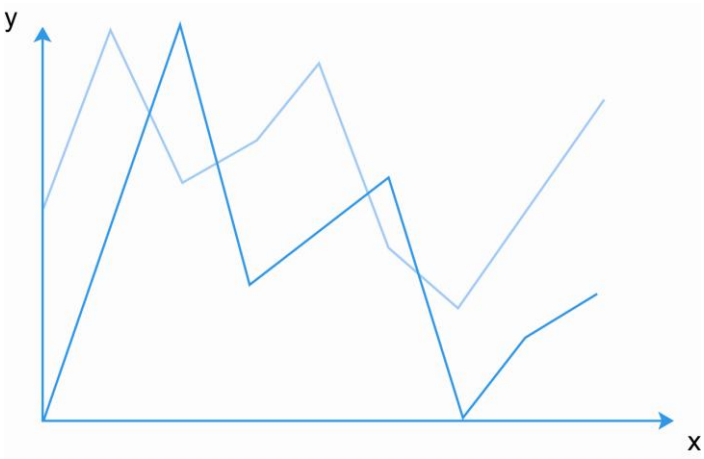


Abb. 8 - Beispiel für ein Liniendiagramm

2.5.3 Torten- oder Kreisdiagramme

In Torten- bzw. Kreisdiagrammen werden Teilwerte (bis ca. 10) eines Ganzen im Verhältnis zueinander und zum Ganzen dargestellt. Hierzu dient die Aufteilung eines Kreises in seine Sektoren, welche durch unterschiedliche Farbgebungen, Muster oder Schattierungen zu differenzieren sind. Variationen von Torten- oder Kreisdiagrammen findet man in Form sogenannter *Halbkreis-* oder *Ringdiagramme*.

Abb. 9 - Beispiel für ein Kreisdiagramm

2.5.4 Thematische Karten

Thematische Karten, auch *angewandte Karten* genannt, visualisieren die Verbreitung eines Merkmals oder mehrerer Merkmale zu einem Thema. Hier werden meist geografische und statistische Daten miteinander verbunden. Bei thematischen Karten, deren Geometrie in den Hintergrund rückt, spricht man von *Kartogrammen*.

Heutzutage finden Infografiken in Form einer Karte überwiegend in sogenannten Geoinformationssystemen (kurz GIS) Verwendung.

Abb. 10 - Beispiel für eine thematische Karte (Deutschland)

3 Zielsetzung der Arbeit

Nach dieser Einführung in die Thematik der Informationsvisualisierung bleibt die Frage offen, wie die angeführten Visualisierungsmethoden hinsichtlich sehr großer Datenmengen auf den Bereich des *Rich Internet* übertragen werden können. Dabei macht die Wahl des geeigneten Frameworks zur Erstellung einer *Rich Internet Application* mit all ihren Anforderungen, die eine neue, noch nicht dagewesene Form der Datenvisualisierung mit sich bringt, eine umfangreiche Recherche im Bereich der RIA-Technologien unausweichlich [vgl. Kap. 4.2.2]. Es gilt abzuwägen, welches Framework, welche Technologie den größten Nutzen liefert, d.h. schnellstmögliche Produktion mit höchst agilem Ergebnis. Welche Features die Anwendung im Einzelnen enthalten soll und welche Faktoren bei der Konzeption und Entwicklung eine wichtige Rolle spielen, ist dem Kapitel der Anwendungsanforderungen [vgl. Kap. 4.2.1] zu entnehmen. An dieser Stelle seien aber schon ein paar Stichworte genannt: Agilität und Benutzerfreundlichkeit gehören genauso dazu wie Interaktivität und hohe Reaktionsgeschwindigkeit. Die Fragen nach dem geeigneten Framework, der Aufstellung eines Entwicklerteams, der Projektplanung, sowohl aus zeitlicher als auch technischer Sicht und der eigentlichen Umsetzung des Projekts werden im folgenden Kapitel der Konzeption herausgearbeitet und beantwortet.

4 Konzeption

Das folgende Kapitel der Konzeption beschäftigt sich zunächst mit der Analyse des Datenbestandes. Darauf aufbauend werden die Anforderungen an die Applikation definiert. Diese Features und Funktionen bilden wiederum die Grundlage für die Wahl der geeigneten Technologie. Zudem beschäftigt sich dieses Kapitel mit den wichtigsten Programmierprinzipien, die ebenfalls die Entscheidung zur Technologie mit geprägt haben.

4.1 Datenbestand

Die für diese Arbeit verwendeten Daten stammen aus den Datenbeständen des Statistischen Bundesamtes Deutschland, welches über den zentralen Datenzugang *GENESIS-Online* statistische Informationen über das Internet zur Verfügung stellt. Neben vorgefertigten Tabellenstrukturen ist es möglich, über drei mögliche Zugangsmodelle (Gastnutzer, Kunde oder Premiumkunde) ausgewählte Datenbestände in einem linearisierten Format zu exportieren. Die Grundlage hierfür stellt das Standardformat *CSV* (engl. character-separated values) dar, wobei es sich um ein Textformat handelt, bei dem die einzelnen Felder (Texte, Werte) durch ein Trennzeichen, im vorliegenden Fall ein Semikolon *";"*, separiert werden.

Aktuell umfasst das Datenangebot über GENESIS-Online folgende Sachgebiete:

- Gebiet, Bevölkerung, Arbeitsmarkt, Wahlen
- Bildung, Sozialleistungen, Gesundheit, Recht
- Wohnen, Umwelt
- Wirtschaftsbereiche
- Außenhandel, Unternehmen, Handwerk
- Preise, Verdienste, Einkommen und Verbrauch
- Öffentliche Finanzen
- Gesamtrechnungen

4.1.1 Datenschnittstelle und Datenmodell

Wie dem Auszug aus Wirtschaft und Statistik des Statistischen Bundesamtes Deutschland "GENESIS-Online Die Internetdatenbank des Statistischen Bundesamtes" von Dr. Claudia Flick [F04] zu entnehmen ist, wird in diesem speziellen Datenmodell eine logische Struktur der Daten- bzw. Wertespeicherung in n-dimensionale Tabellen, hier *Datenquader* genannt, gewählt. Dieses Modell baut darauf auf, dass alle zu einem Datenquader gehörigen Informationen zu einer vollständigen Dokumentation zusammengeführt werden können. Dies erfolgt über die Referenzierung zu einer Metadatensammlung, welche Angaben über den sachlichen Hintergrund und die inhaltliche Aussage einer Zahl enthält. Wie in der folgenden Abbildung verdeutlicht wird, umfasst ein Datenquader die Daten, welche zu genau einer Statistik gehören.

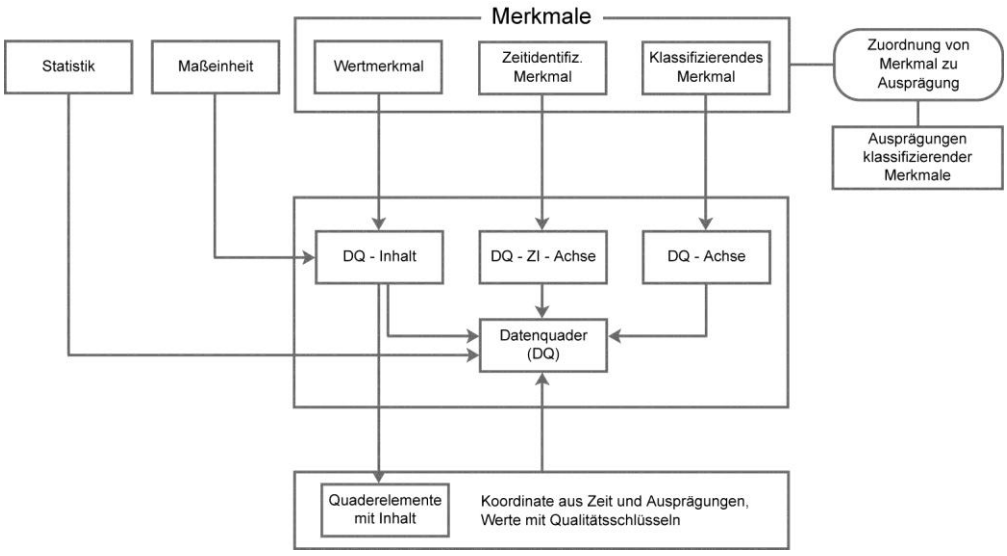


Abb. 11 - Das Datenmodell von GENESIS-Online im Überblick

Der sachliche, räumliche und zeitliche Bezug der Daten wird über die Angabe der entsprechenden klassifizierenden Merkmale (K-Merkmale) und des zeitidentifizierenden Merkmals (ZI-Merkmal) hergestellt. Im logischen Datenmodell definieren diese Merkmale die Achsen des Quaders (DQ-Achsen, DQ-ZI-Achse). Über die Angabe eines Wertmerkmals als Datenquaderinhalt (DQ-Inhalt) wird dokumentiert, welche statistische Angabe im Quader wertmäßig gespeichert ist. Es können mehrere Wertmerkmale in einem Datenquader gespeichert

werden. Den im Datenquader wertmäßig erfassten Angaben (Datenquaderinhalten) ist jeweils eine Maßeinheit zugeordnet.

Beim Export der Daten sind logische Abhängigkeiten von großer Bedeutung. Dieser erfolgt in einer fest definierten Reihenfolge, die aufgrund der Abhängigkeiten zwischen Meta- und Wertedaten notwendig ist. Es ist zwischen dem Import von Metabeschreibungen und dem Import der eigentlichen Werte zu unterscheiden. Der Aufbau einer gewünschten Tabelle wird über die Anordnung der Metadaten in einer sogenannten *Abruftabelle* definiert. Dies ist eine Ordnungsstruktur, in der die Inhalte der einzelnen Strukturelemente einer Tabelle festgelegt werden. Zu diesen Strukturelementen zählen Kopfzeile, Vorspalte, Untertitel, Zwischentitel sowie globale Angaben, die für die ganze Tabelle gültig sind. Entsprechend den Vorgaben der Abruftabelle wird eine Ergebnistabelle, also eine mit Werten gefüllte Tabelle, erzeugt. Der Vorteil des Quadermodells besteht darin, sowohl sachlich tief gegliederte Strukturdaten als auch Regionaldaten, Verknüpfungen sachlicher Art und auch Zeitreihen bereithalten und für den Direktzugriff anbieten zu können.

4.1.2 Tabellen

GENESIS-Online ermöglicht dem Nutzer laut des Auszugs aus Wirtschaft und Statistik [F04] über flexible Tabellenstrukturen auf unterschiedliche Daten zugreifen zu können. Dabei können die Ausprägung von vorgegebenen Merkmalen sowie die Merkmale für eine bestimmte Tabellenposition selbst ausgewählt werden. Zu variieren sind beispielsweise der gewünschte Zeitraum oder die Auswahl von Positionen aus fachlichen und regionalen Gliederungen [vgl. Abb. 12]. Durch eine Vorschau kann der individuell modifizierte Tabellenrahmen angezeigt werden. Der Werteabruf kann sowohl nach vorheriger Modifikation als auch unmittelbar nach dem Aufruf einer Tabelle mit ihren Standardeinstellungen erfolgen. Die Ergebnistabelle enthält immer den aktuellen Stand der Statistik und steht dem Nutzer in den Formaten HTML, CSV und XLS (Microsoft Excel) zum Download zur Verfügung [vgl. Abb. 13].

Aber nicht nur die eigentlichen Daten sondern vor allem auch die zugehörigen Metadaten sind genau dann von Bedeutung, wenn diese Daten in eigenen Datenbanken weiterverwendet werden sollen.

Abb. 12 - Tabellenaufbau GENESIS-Online

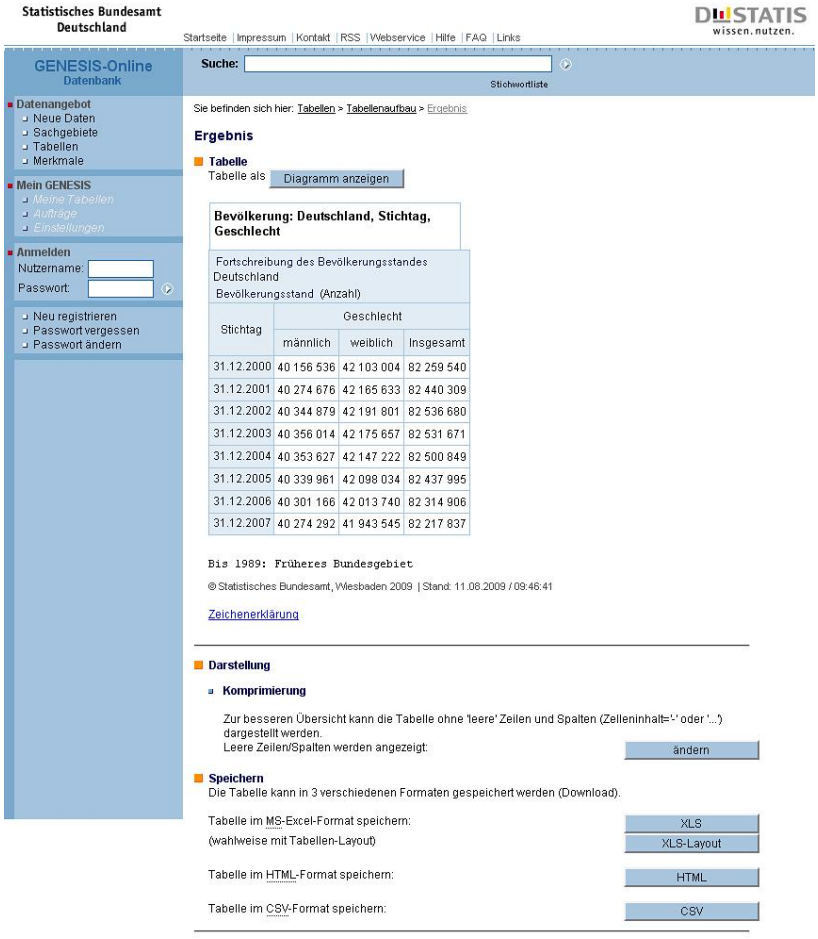


Abb. 13 - Ergebnistabelle GENESIS-Online

4.2 Anwendungskonzeption

Zur Beantwortung der Fragestellung nach der richtigen Technologie oder dem richtigen Framework zur Software-Entwicklung muss zunächst klar definiert werden, was die geplante Anwendung an Funktionalität erhalten soll. Hierbei ist es wichtig, zu berücksichtigen, wie sich die Applikation weiter entwickeln könnte.

4.2.1 Anwendungsanforderungen

Wie dem Thema der vorliegenden Diplomarbeit zu entnehmen ist, soll eine Rich Internet Application zur interaktiven Visualisierung von Massendaten entwickelt werden. Bereits im vorherigen Kapitel wurde erläutert, dass sich diese Arbeit auf die Datengrundlage des Statistischen Bundesamtes Deutschland bezieht, das über die Internetschnittstelle GENESIS-Online Zugriff auf dessen Datenbank gewährt. In welcher Form diese Daten vorliegen und wie sie weiterverarbeitet werden können, ist bereits aus Kapitel 4.1 bekannt. Jedoch gilt es an dieser Stelle zu bedenken, dass die Applikation mit dem Codenamen *Zwickr* zu späterer Zeit ein Interface für mögliche weitere Datenbestände darstellen soll. Das zu planende Datenmodell ist demnach möglichst agil zu halten.

In erster Linie wird eine plattformübergreifende Anwendung angestrebt, also eine Applikation, die auf den Betriebssystemen Windows und Mac OS, evtl. aber auch auf Linux läuft, vorausgesetzt das nötige Browser-Plugin ist clientseitig installiert.

An das Design der Anwendung werden hohe Erwartungen gestellt. Es soll sich von vergleichbaren Applikationen abheben und dem Benutzer einen möglichst intuitiven Zugang zu den statistischen Daten bieten. *Look and Feel* und Nutzen müssen sich dabei in Einklang befinden und die Hauptapplikation somit zu einem Alltagswerkszeug werden lassen. Zwickr soll somit nicht bloß ein Abbild einer statistischen Datenbank in attraktivem Design werden, sondern vielmehr zusätzlich die Möglich-

keit der gezielten Suche nach konkreten Daten bieten. Die Erfahrung mit GENESIS-Online zeigt, wie verschachtelt dort Daten hinterlegt sind und wie viele Schritte vonnöten sind, ehe die gewünschte Tabelle erreicht ist. Zudem kann eine Tabelle aus mehreren hundert Zeilen und vielen Spalten bestehen, wodurch die schnelle Informationsnutzung erschwert wird. Der aktuelle Stand des statistischen Informationsangebots im Internet zeigt, welch komplexe Logik hinter so einer Datenquelle steckt, leider aber auch, dass der Zugang zu diesen Daten bislang nicht wirklich interessant umgesetzt wird. Genau an dieser Stelle setzt diese Diplomarbeit ein. Ziel ist es, eine Schnittstellenapplikation am Beispiel des Statistischen Bundesamtes Deutschland zu entwickeln, welche dem Benutzer über eine intuitive Navigation den Weg zu konkreten Daten ebnet. Er soll die Möglichkeit erhalten, während einer Recherchephase für ihn interessante Themen auswählen und in einer Sammlung temporär ablegen zu können. Er erhält zu den einzelnen Themen die Möglichkeit, differenziertere Parametereinstellungen vorzunehmen. Die angezeigten Informationsgrafiken sind interaktiv an bestimmte Parameter gebunden, so dass jederzeit in die Darstellung eingegriffen werden kann. In besonderen Fällen werden Abhängigkeiten zwischen verschiedenen Datensätzen dadurch visualisiert, dass verschiedene Diagrammformen zusammen in einer Übersicht, der sogenannten *Multibox* angezeigt werden und interaktiv miteinander verknüpft sind. Der Mehrwert dieser Datenvisualisierung über eine RIA im Vergleich zum aktuellen Stand der Informationsvisualisierung im Internet ist beträchtlich und kann in unterschiedlichen Nutzerszenarien zukünftig von großer Bedeutung sein.

4.2.2 Die Wahl des RIA-Frameworks

Bevor auf die Entwicklung einer Enterprise-Webanwendung näher eingegangen wird, sind einige planungstechnische und konzeptionelle Schritte zu beachten. Eine sorgfältige Planung wird dem Projektteam zu späterer Zeit viel Ärger ersparen, bedenkt man von Beginn an die Skalierbarkeit der Anwendung sowie die Grundsätze einer erfolgreichen Programmierung (*Best Practice Management*), wie beispielsweise die *lose gekoppelte* bzw. *Ereignisgetriebene* oder *Objektorientierte Programmierung* [vgl. Kap. 4.2.3].

4.2.2.1 Was ist eine RIA?

Der Begriff *Rich Internet Application* wurde 2002 von Macromedia geprägt und bedeutet wörtlich übersetzt *Reiche Internet Anwendung*, meint aber vielmehr die *Reichhaltige Internet Anwendung*. Mittlerweile werden schon Vermutungen angestellt, dass RIAs auch als *nächste Generation von Browseranwendungen* bezeichnet werden könnten. Harry Klein schafft in seinem Artikel für das Magazin *Create Or Die* einen Überblick [CoD08]:

Bei dem Innovationsgedanken stehen im Wesentlichen eine verbesserte Benutzerfreundlichkeit und eine höhere Intelligenz der Applikation im Vordergrund. Gegenüber herkömmlichen, HTML-basierten Internetseiten bestehen hier grundsätzlich neue Softwareansätze, die der Benutzer von reinen Desktop-Anwendungen her kennt. Als Beispiel hierfür sind unter anderem Drag & Drop- und Multimedia-Funktionalitäten, d.h. die Einbettung von Audio- und Video-Daten, zu nennen. Abgesehen von der Benutzeroberfläche bieten RIAs zudem eine hohe Leistungsfähigkeit, auf Benutzeranfragen schnell und performant reagieren zu können. Dieser Vorteil gegenüber traditioneller Webapplikationen ist darauf zurück zu führen, dass der Client, als Mittler zwischen Browser und Server, den größten Teil der Programmlogik übernimmt. Er ist verantwortlich für das Applikationsinterface und die asynchrone Kommunikation zum Server, die die Reaktionsgeschwin-

digkeit und Performance der Web-Anwendung enorm erhöht. Somit ähnelt eine RIA immer mehr einer Desktop-Applikation, mit dem erheblichen Vorteil, dass diese nicht zuerst installiert werden muss.

4.2.2.2 RIA-Technologien – Ein Überblick

Im Bereich der RIA-Entwicklung gibt es bereits seit einigen Jahren verschiedene Alternativen, welche alle bestimmte Vor- und Nachteile aufweisen und sich somit, je nach Bedarf, mehr oder weniger gut für die Erstellung von Enterprise-Anwendungen eignen. Der Ursprung der RIA-Technologien liegt in der AJAX-Technologie, (Asynchronous JavaScript and XML). Wie der Titel schon beschreibt, dient diese Technologie der asynchronen Kommunikation zwischen Client und Server, d.h. AJAX-Anwendungen können Anfragen an einen Server richten, bei denen nur die Daten angefordert werden, die tatsächlich benötigt werden. Dies geschieht über den Aufruf eines Web-Services wie REST (Representational State Transfer) oder SOAP (Simple Object Access Protocol). Während die Daten vom Server geladen werden, kann der Benutzer weiterhin mit der Oberfläche agieren. Nach abgeschlossenem Ladevorgang der Daten wird eine zuvor benannte JavaScript-Funktion aufgerufen und die Daten in die Web-Anwendung eingebunden. [W09a]

Zwei äußerst beeindruckende Frameworks zur Entwicklung von RIAs wurden am 28. März 2009 beim 25. Multimediatreff in Köln unter dem Titel „Flash, Flex & Silverlight – reichlich Web!“ vorgestellt. Die dort entstandenen Videobeiträge stehen im Internet zur freien Verfügung:
`http://www.galileodesign.de/techtalks#flashflexair`

Zum einen handelte es sich bei den präsentierten Frameworks um das von der Microsoft Corporation angebotene Silverlight, welches ursprünglich den Namen *WPF/E* (Windows Presentation Foundation/ Everywhere) trug. Zum anderen wurde das einst von Macromedia ins Leben gerufene Flex-Framework, welches inzwischen von Adobe Systems übernommen und weiterentwickelt wurde, präsentiert. Beide Frameworks sind mittlerweile, aktuell jeweils in der dritten Generation verfügbar, zu mächtigen Konkurrenten herangewachsen, deren

Entwickler und Herausgeber sich vermutlich im Duell um die RIA-Technologie der Zukunft immer wieder gegenüber stehen werden. Hierzu stellen Kai König und John-Daniel Trask die beiden RIA-Werkzeuge in ihrem iX-Artikel gegenüber [Ho08]:

Während Silverlight in der ersten Version 2007 veröffentlicht wurde, gab es Flash- und Flex-Anwendungen bereits ein paar Jahre zuvor. Es sollte sich als Alternative zu den Adobe-Produkten etablieren und wies erstaunlich viel Ähnlichkeit mit dem Konkurrenz-Framework auf – was sich bis heute nicht grundlegend geändert hat.

Für die Nutzung einer in Silverlight oder Flex entwickelten Applikation ist in beiden Fällen ein Browser-Plug-in notwendig. Beide Frameworks bedienen sich einer XML-basierten Markup-Sprache, XAML (Silverlight) und MXML (Flex). Zunächst hatte Adobe - später dann aber auch Microsoft - sich zum Ziel gesetzt, mit ihren Frameworks eine integrierte Entwicklungsumgebung (engl. Integrated Development Environment (kurz IDE)) zu schaffen, die sowohl den Anspruch von Designern als auch von Entwicklern erfüllt. Während auf den ersten Blick beide Framework-Modelle sehr ähnlich erscheinen, besteht ein grundlegender Unterschied. Silverlight-Entwickler können sowohl den Client als auch Serveranwendungen auf Basis von Microsofts CLR in der .NET-Sprache ihrer Wahl entwickeln. Dahingegen müssen Flex-Entwickler strikt zwischen Client und Server trennen: Ihre Verantwortung reicht lediglich bis zur Schnittstelle, an der Anwendungsdaten zwischen Client und Server übergeben werden. Allerdings wird diese von der .NET-Community häufig als Nachteil empfundene Gegebenheit durch eine Vielfalt von Backend-Techniken, die an die Flash-Platform angebunden werden können, ausgeglichen. So existiert neben XML-Webservices und HTTP-Aufrufen ein Remoting-Verfahren, welches über sogenannte `RemoteObject`-Klassen in Flex genutzt werden kann [vgl. Kap. 5.3.3]. Darüber hinaus können mithilfe eines serverbasierten Remoting Gateways problemlos Plattformen wie Java, .NET, PHP und viele andere an Flex und Flash angebunden werden.

Ein abschließender Vergleich der beiden Frameworks auf ihre jeweiligen Klassen- und Komponentenbibliotheken bezogen zeigt, dass Adobes Flex-Framework bislang den Vergleich deutlich gewinnt. Speziell die UI-Komponenten von Flex 3 sind ausgereifter und vielfältiger als die Oberflächen-Komponenten von Silverlight. Weitere bestehende Nachteile der Silverlight-Technologie sind eine fehlende Druckfunktion aus der Applikation und die eingeschränkte Möglichkeit, lediglich auf PC-Basis entwickeln zu können. Ebenso sprechen die Zahlen der Plug-in-Verbreitung [vgl. Tab. 1] und die Größe der Entwicklergemeinden hinter den beiden Konkurrenten dafür, dass bislang die RIA-Technologie von Adobe mit Flash und Flex uneingeholt bleibt.

Aufgrund dessen wird die geplante Anwendung mit der Technologie von Flash und Flex realisiert.

	Flash Player 7	Flash Player 8	Flash Player 9	Flash Player 10
Mature Markets	99.0 %	98.9 %	98.8 %	86.7 %
US/ Canada	99.2 %	99.2 %	99.1 %	87.2 %
Europe	98.6 %	98.5 %	98.1 %	85.8 %
Japan	98.8 %	98.8 %	98.8 %	86.8 %
Emerging Markets	97.9 %	97.7 %	97.1 %	81.8 %

Tab. 1 - Weltweite Nutzung des Adobe Flash Players nach Version

4.2.3 Best Practice Management

Das folgende Kapitel beschreibt und erläutert ausführlich das Prinzip des *Best Practice Management* und stützt sich dabei auf die Seminarinhalte des Fachs Softwareentwicklung der Johannes Kepler Universität Linz (WS2002/03) [S09]:

Das *Best Practice Management*, zu übersetzen mit *beste Vorgehensweise* oder *Erfolgsmethode*, stammt aus der angloamerikanischen Betriebswirtschaft. Bei diesem Prinzip geht es im Allgemeinen darum, bewährte, kostengünstige Techniken, Technologien und Managementsysteme einzusetzen, um sich dadurch als Musterunternehmen hervorzuheben [W09b].

Der Bereich des Best Practice Management in der Softwareentwicklung setzt sich aus sechs Faktoren zusammen:

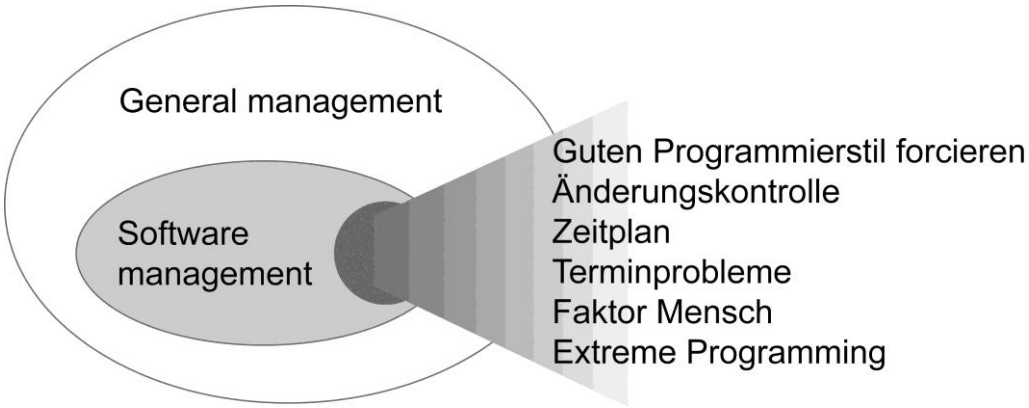


Abb. 14 - Best Practice Management in der Softwareentwicklung

Guten Programmierstil forcieren

Wie kann guter Programmierstil erreicht werden?
Nicht immer führen strikte Standards zu gewünschten Ergebnissen. Gerade bei der Arbeit im Team können starre Regeln zu Konflikten führen, da stilistische Fragestellungen wie z.B. „Setze ich die Klammer in der selben oder in der nächsten Zeile?“ persönliche Präferenzen sind. Aus diesem Grunde gibt es verschiedene Lösungsansätze, welche miteinander kombiniert oder jeder für sich das Ziel verfolgen, guten Programmierstil zu forcieren. Auch wenn sich kein weltweiter Standard zum einheitlichen Programmieren einführen lässt, ist es von Vorteil, innerhalb eines Teams einen gemeinsamen Standard zu nutzen. So lässt sich ein gemeinschaftlicher Code, sei er zu zweit oder in größeren Gruppen erarbeitet, permanent kontrollieren und beurteilen. Nicht selten ist es zudem üblich, Programmsequenzen von einer erfahrenen Person, beispielsweise dem Vorgesetzten, abzeichnen zu lassen. Besonders gute Codesequenzen führen bei einer Präsentation im Team dazu, dass sich jedes Teammitglied anhand des Beispiels schnell über guten Programmierstil bewusst wird, ohne auf lästige Formalismen zurückgreifen zu müssen. Auch wenn ein Code als persönliches Eigentum gesehen werden kann, profitieren andere Entwickler von Positiv- oder Negativbeispielen, wenn er der Öffentlichkeit zugänglich gemacht wird. So sind einschlägige Entwicklerforen im Internet der ideale Ort, um zu lernen oder zu lehren.

Änderungskontrolle

Änderungskontrolle ist die deutsche Übersetzung des englischen Fachbegriffs *Software Configuration Management* (kurz SCM).
Wie kann das Bestreben der Änderungskontrolle, Projekte immer konsistent zu halten, erreicht werden?
In den einzelnen Phasen eines Projektes können permanent Änderungen auftreten. Je nach Projektphase können demnach weitere Änderungen notwendig sein. Änderungen in den Anforderungen haben Auswirkung auf das Design, welches wiederum Codeerzeugung und Dokumen-

tation beeinflusst. Dies zeigt sehr offenkundig, wie komplex ein solcher Prozess sein kann und wie fehleranfällig die Anwendung dadurch wird. Handelt es sich um eine Designänderung, ist anzuraten, Änderungsprozesse schon im Vorfeld formal zu beschreiben, so dass jeder Beteiligte das Vorgehen bei einer Änderung verstehen und nachvollziehen kann. Jede Änderung bringt einen gewissen Aufwand mit sich, der im Vorfeld abgeschätzt und kalkuliert werden muss. Selbst wenn es sich nur um eine kleine Änderung handelt, kann der zu betreibende Aufwand enorm sein. Ein unausgereiftes Anforderungskonzept, wie beispielsweise das Lastenheft [vgl. Kap. 4.3.4], kann zu häufigen Änderungswünschen seitens des Auftraggebers führen. In diesem Fall sollte eine Nachbearbeitung des Lastenheftes unbedingt erfolgen, denn jede postume Änderung würde einen Mehraufwand an Zeit und Geld bedeuten.

Als Unterstützung bei Quellcodeänderungen eignen sich sogenannte *Version-Control-Systeme*. Die Funktion dieser kann man sich in etwa so vorstellen, dass eine zur Bearbeitung aufgerufene Datei mit einer Markierung versehen wird (check out), die erst nach der Bearbeitung wieder aufgehoben wird (check in). Der Entwickler hat zudem die Möglichkeit, zusätzlich einen Erklärungstext anzufügen. Bei jedem *check in* wird die alte Datei nicht überschrieben, so dass eine Änderungshistorie entsteht, auf die jederzeit zurück gegriffen werden kann.

Kosten- und Zeitplan

„Im Durchschnitt ist ein Softwareprojekt ein Jahr verspätet und das Budget um 100 Prozent überzogen“ [S09]. Wie können Kosten und Zeit für ein anstehendes Projekt richtig eingeschätzt und geplant werden?

Zur Kostenschätzung kann ein algorithmisches Kostenschätzungsmodell, wie z.B. *COCOMO* (*Constructive Cost Model*), genutzt werden. Dieses basiert auf der Analyse von Daten aus vergangenen Projekten. Zum einen stellt das Modell eine Beziehung zwischen der Größe des zu erzeugenden Software-Produktes und dem Aufwand in Personenmonaten dar. Zum anderen lassen sich 16 Kostenfaktoren (engl. Cost

Drivers) aus vier Kategorien (Produkt, Hardware, Personal und Projekt) multiplikativ in den Aufwand hineinberechnen.
Da Kostenschätzungsmodelle generell auf Erfahrungswerten beruhen, ist es essentiell, dass Daten über vergangene Projekte erhoben wurden. Der Zeitbedarf wird aus der Aufwandsabschätzung abgeleitet.

Damit Fehlkalkulationen, die zu Beginn eines Projekts entstanden sind, nicht während des gesamten Projekts zu fehlerhaften Folgekalkulationen führen können, ist es ratsam, ein Projekt in der Planungsphase in kleinere Teile, so genannte *Meilensteine* (engl. Milestones) zu zerlegen und diese anschließend zu bewerten. Dies bedeutet zwar für die Anfangsphase eines Projektes einen gewissen Mehraufwand, jedoch rentiert sich dieser innerhalb der gesamten Projektlaufzeit. Im Hinblick auf die Aufwands- und Kostenkalkulation ist es wichtig, die einzelnen Projektbeteiligten einzubeziehen, indem man sie den Aufwand für ihre Projektteile selbst einschätzen lässt.
Auch im Bereich der Zeitplanung gibt es keine einheitliche Lösung. Jede Methode hat gewisse Vorteile, die am besten durch die Verbindung mit anderen Methoden zum Tragen kommen. Zudem wirkt sich eine permanente Beurteilung des Projektverlaufs positiv auf die Kalkulation aus.
Wie die folgende Grafik zeigt, steigt der Aufwand für ein Projekt nicht proportional mit der Projektgröße. Je größer ein Projekt ist, desto mehr Zeit ist in das Design, die Architektur und die Testphase zu investieren, weniger aber in die eigentliche Implementierung.

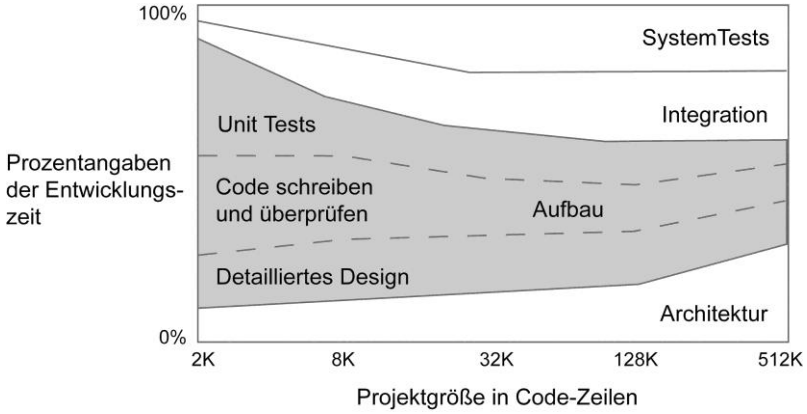


Abb. 15 - Abhängigkeit des Aufwands für die Projektphasen von der Projektgröße

Terminprobleme

Wie ist der Problematik zu begegnen, wenn das Projekt offensichtlich nicht termingerecht fertig wird?
Auch wenn das vorangegangene Kapitel Wege der Aufwandseinschätzungen aufzeigt, ist es nicht selten, dass eine Kalkulation um rund 30% zu optimistisch ausfällt. Da diese Problematik meist erst zum Ende einer Projektlaufzeit auffällt, lässt es sich selten vermeiden, dass im Bereich der Funktionalität Einsparungen vonnöten sind. Um solche Terminprobleme zu vermeiden, sollte eine genaue Terminplanung und Aufwandseinschätzung bereits im Vorfeld geschehen.

Faktor Mensch

Wie ist ein Team für ein Projekt im Bereich der Softwareentwicklung unter dem Aspekt der Produktivität zusammenzustellen? Da Software von Menschenhand produziert wird und Personalkosten den Hauptanteil der Produktionskosten ausmachen, sollte neben dem Produktionsfaktor Geld der Faktor Mensch im Rahmen der Aufwandseinschätzung besonders fokussiert werden.
Für jedes Team, unabhängig von der Gruppengröße, ist die interne Kommunikation und Organisation äußerst wichtig. Dieser nichtproduktive Aufwand nimmt mit zunehmender Personenzahl überproportional zu. Veränderungen im Team, wie beispielsweise durch Hinzukommen oder Ausfallen eines Entwicklers, werden die Schwierigkeiten des
Organisierens eines Teams zusätzlich verstärkt, bzw. der Entwicklungsprozess vorübergehend verlangsamt.

Extreme Programming (xProgramming)

Wie kann eine hohe Softwarequalität und Kundenzufriedenheit erreicht werden?
Eine neuartige Variante der Software-Entwicklung, das *Extreme Programming* (kurz XP), wurde 1996 durch Kent Beck, Ward Cunningham und Ron Jeffries für den Automobilkonzern Daimler Chrysler

entwickelt [W09c]. Die Idee des Extreme Programming befasst sich mit den sich häufig ändernden Anforderungen innerhalb des Entwicklungszyklus, auf welche mit klassischen Vorgehensmodellen nicht optimal reagiert werden kann. Kommunikation ist dabei der wichtigste Bestandteil dieser Methode. Es entsteht ein Produkt, an dessen Entwicklung der Kunde aktiv teilgenommen hat. Innerhalb des Entwicklerteams werden neue Funktionalitäten permanent entwickelt, integriert und getestet. Auf dem Weg der zu entwickelnden Funktionalität sind die Schritte der Risikoanalyse, der Nutzenanalyse, der Bereitstellung einer ersten ausführbaren Version (Prototyping) sowie des Akzeptanztests zu durchlaufen [vgl. Abb. 16].

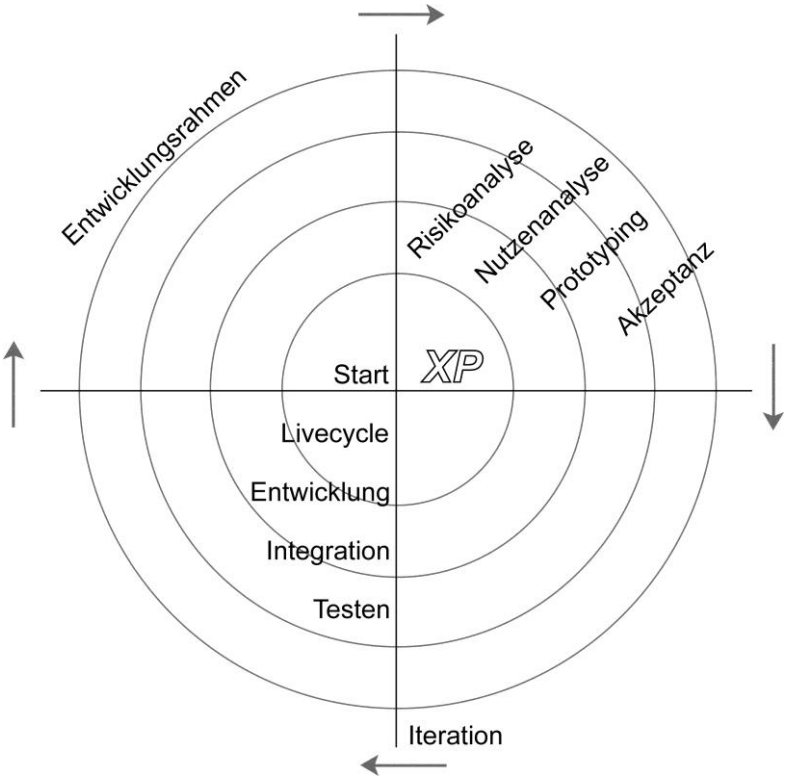


Abb. 16 - XP Lebenszyklus

Aus Kundensicht bietet Extreme Programming die Möglichkeit, jederzeit steuernd auf das Projekt einzuwirken. Bestenfalls befindet sich der Kunde zu jeder Entwicklungsphase des Projekts auf demselben aktuellen Informationsstand wie das Entwicklerteam. Die ständige, aktive Einbeziehung des Kunden soll zudem der sogenannten *Featuritis*

vorbeugen. Dies ist ein nicht seltenes Problem in der Softwareentwicklung, bei dem Programmfunktionen erstellt werden, die vielleicht, jedoch nicht mit Sicherheit, eines Tages gebraucht werden könnten. Im Gegensatz dazu wird bei Extreme Programming eindeutig definiert, was sinnvoll ist zu entwickeln.

Aus Sicht der Entwickler sieht Extreme Programming keine strikte Rollentrennung der Teammitglieder vor. So gibt es im Team kein Wissensmonopol. Die Aufgabenverteilung erfolgt situationsbedingt und in Abhängigkeit von personellen Fähigkeiten und Fertigkeiten. Dies macht sich in der Entlastung des Einzelnen positiv bemerkbar, da jedes Teammitglied einen ähnlichen Wissensstand aufweist und bestimmte Problemlösungen nicht im Kompetenzbereich einer einzelnen Person liegen. Fehlerhafter Programmierung sowie fehlerhafter Integration von Komponenten beugen Modultests, sogenannte *Unittests*, vor. Erzielt wird eine erhebliche Qualitätssteigerung durch frühes Auffinden von fehlerhaftem Code und permanenter Kontrolle des bereits existierenden und neu hinzukommenden Quellcodes. Vor allem im Team sorgt die sogenannte *testgetriebene Entwicklung* für mehr Kontrolle und Sicherheit während der gesamten Projektlaufzeit.

Auf das Projekt bezogen dient Extreme Programming der Minimierung von Risiken, die während einer Projektarbeit auftreten können. Durch permanente Kommunikation innerhalb des Teams sowie mit den Kunden und andauernder Prioritätsanalysen soll erreicht werden, dass das Produkt in der gewünschten Zeit, im gewünschten Umfang und mit den geplanten Ressourcen fertig gestellt werden kann.

Die Aufgabenverteilung der einzelnen Beteiligten wird in der folgenden Abbildung beispielhaft erläutert:

Rolle	Beispiel	Aufgabe
Produktbesitzer	Produktmanagement, Marketing, ein Benutzer, Kunde, Manager des Benutzers, Analyst, Sponsor	Hat Verantwortung, setzt Prioritäten, entscheidet das beste ROI (Return on Investment)
Kunde	Auftraggeber, kann auch der Produktbesitzer sein, kann, muss aber nicht der Benutzer sein	Entscheidet, was gemacht wird, gibt regelmäßig Rückmeldung, ist Auftraggeber
Entwickler	Bestandteil des Teams, das ganze Entwicklungsteam besteht aus Entwicklern: Programmierer, Tester, DB-Experten, Architekten, Designer	Entwickelung des Produktes
Projektmanager	Ist für gewöhnlich der Produktbesitzer, kann auch Entwickler aber nicht Manager des Teams sein	Führung des Teams
Benutzer	Der Nutzer des zu erstellenden Produktes	Wird das zu erstellende Produkt nutzen

Tab. 2 - Die Rollen bei XP

4.2.3.1 Objektorientierte Programmierung

Bei einem Blick hinter die Benutzeroberflächen von RIAs bemerkt man einen deutlichen Unterschied zu klassischen HTML-Seiten. Bereits seit vielen Jahren hat sich das Programmierparadigma *Objektorientierte Programmierung* (kurz OOP) etabliert. Das gegenteilige Paradigma der *Prozeduralen Programmierung*, bei dem ein Programm, wie der Name schon vermuten lässt, von Anweisung zu Anweisung voranschreitet, ist inzwischen gänzlich abgelöst. Hierzu folgt eine Erläuterung des OOP-Prinzips anhand des Galileo <openbooks> "Objektorientierte Programmierung" von Bernhard Lahres und Gregor Rayman [LR09]:

Entwickelt wurde die OOP unter anderem in Anlehnung an die Art des menschlichen Denkens. Hierbei stehen Objekte im Fokus, die sowohl Daten (Attribute), als auch die Daten verarbeitenden Funktionen

(Methoden) vereinen. So kann eine umfangreiche Anwendung auf viele Objekte aufgeteilt werden - vergleichbar mit der Zusammenarbeit eines Teams - womit die Überschaubarkeit komplexer Anwendungen immens gesteigert werden kann. Innerhalb des Entwicklungsprozesses, also von der *Objektorientierten Analyse* (kurz OOA) über das *Objektorientierte Design* (kurz OOD) bis hin zur *Objektorientierten Programmierung* (kurz OOP), gelten die gleichen Entwurfs- und Dokumentations-Methoden (z.B. Klassen- und UML-Diagramme). Flexibilität wird unter anderem durch das Prinzip der Vererbung erreicht, durch welches bereits entwickelte Klassen an neue Situationen angepasst und wieder verwendet werden können. Voraussetzung hierfür ist ein möglichst allgemeiner Entwurf dieser Klassen.
Neben der Vererbung existieren zwei weitere Grundelemente der Objektorientierten Programmierung: Die *Datenkapselung* und die *Polymorphie*. Die Datenkapselung sorgt dafür, dass Daten in der OOP explizit einem Objekt angehören und nur über dieses auf sie zugegriffen werden kann. Die Polymorphie, übersetzt mit *Vielgestaltigkeit*, erlaubt es, einzelne Elemente gegen andere auszutauschen, womit eine höhere Wartbarkeit und Änderbarkeit der Software gewährleistet wird.

4.2.3.2 Lose gekoppelte Programmierung

Das Prinzip der *losen Kopplung* stammt aus der Informatik und impliziert einen geringen Grad an Abhängigkeiten mehrerer Hard- oder Software-Komponenten untereinander [W09e].

In der Software-Architektur hat sich das Ziel der losen Kopplung ebenfalls etabliert. Sie verfolgt das primäre Ziel, einzelne Komponenten einer Anwendung so eigenständig wie möglich zu entwickeln, so dass auf sie über klar definierte Schnittstellen jederzeit zugegriffen werden kann, ohne den Inhalt der Komponente im Detail kennen zu müssen – das sogenannte *Black Box Prinzip*. Dies funktioniert beispielsweise über das Feuern von Events, auch Ereignisgetriebene Architektur (engl. Event-Driven Architecture (kurz EDA)) genannt. Events werden in diesem Fall nicht direkt auf die Empfängerkomponente gefeuert,

sondern von einer Vermittlerschicht abgefangen, die über entsprechende Schnittstellen (Events und öffentliche (`public`) Methoden) die Kommunikation zwischen den einzelnen Komponenten übernimmt. Ändert sich eine Komponente, muss diese Veränderung nur an einer zentralen Stelle, der Vermittlerschicht vorgenommen werden, was ein klarer Vorteil dieser Vorgehensweise ist. [W08a]

Die folgenden zwei Schaubilder verdeutlichen den Unterschied zwischen loser und enger Kopplung:

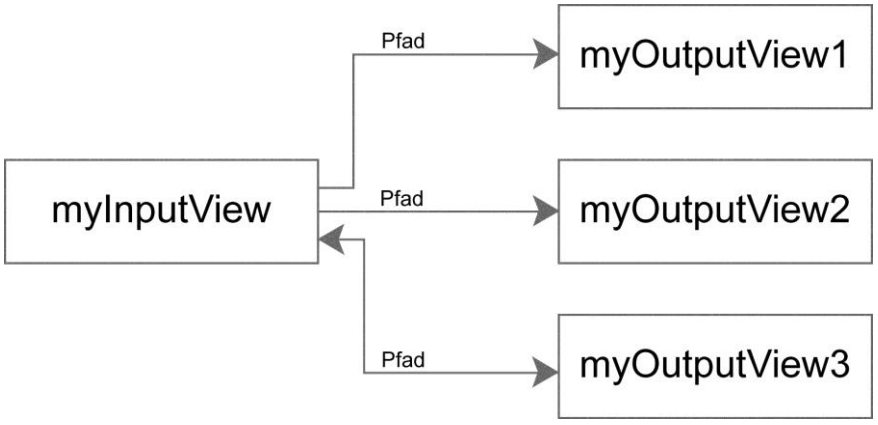


Abb. 17 - Enge Kopplung (engl. tight coupled)

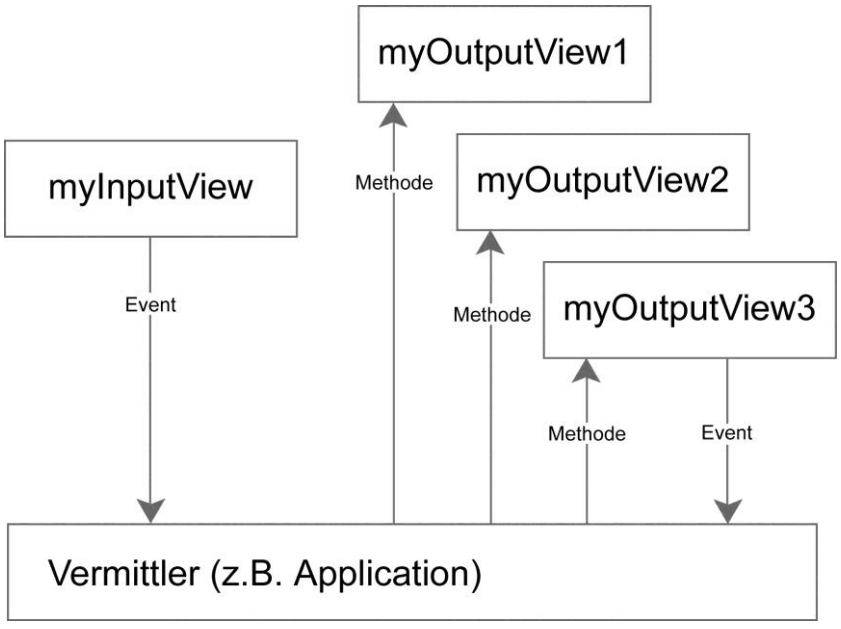


Abb. 18 - Lose Kopplung (engl. loosely coupled)

4.2.3.3 Model View Controller (MVC)

Um eine Enterprise-Anwendung agil und wartbar zu entwickeln, sollte zu Beginn ein wenig Zeit in die Planung einer gelungenen Projektarchitektur investiert werden. Das für flexible Programmierung entwickelte *Model-View-Controller*-Konzept (kurz MVC) beschreibt in der Software-Entwicklung die Zusammenhänge und strikte Trennung der drei Einheiten: Datenmodell (engl. Model), Präsentation (engl. View) und Programmsteuerung (engl. Controller). Das Prinzip dieses Entwurfsmuster (engl. Design-Pattern) liegt in der Flexibilität der einzelnen Einheiten, womit spätere Änderungen oder Erweiterungen erleichtert werden, bzw. eine Weiterverwendung einzelner Module ermöglicht wird [W09f].

Bei näherer Betrachtung lassen sich die drei Hauptbestandteile dieses Konzepts wie in folgender Abbildung beschreiben:

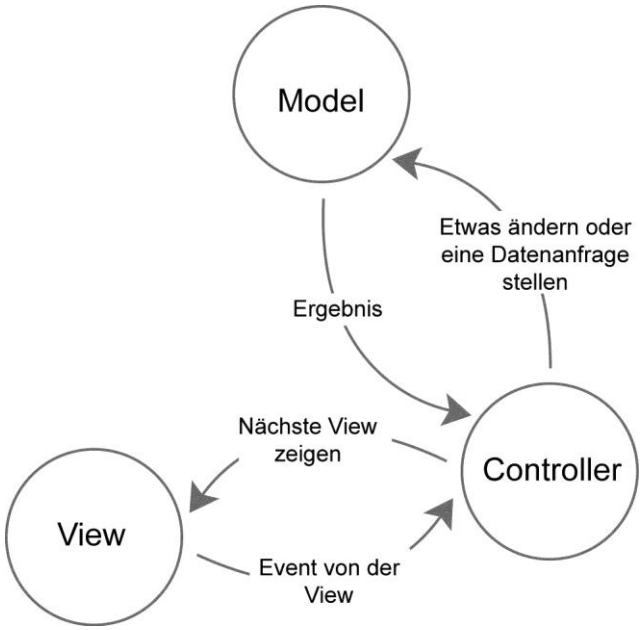


Abb. 19 - Model-View-Controller-Konzept

Model

Das Model ist ausschließlich für die Datenhaltung, nicht aber für die Visualisierung oder Logik zuständig.

View

Die View stellt die grafische Benutzeroberfläche dar. Sie ist für die Darstellung der Daten in dem jeweils durch die Parameter konfigurierten Model zuständig.

Controller

Der Controller überwacht als Steuerungseinheit alle Benutzereingaben und ist für die Umsetzung der eigentlichen Programmlogik zuständig.

Im Allgemeinen erfolgt die Kommunikation zwischen Model und View über ein *Observe-Design*-Pattern, wobei eine Veränderung im Model sofort registriert und an alle angebundenen Views gesendet wird (beispielsweise über eine Datenbindung), so dass diese sich automatisch aktualisieren.

Vorteil des MVC-Konzepts ist beispielsweise die Austauschbarkeit einer oder mehrerer Views durch eine neue View auf das bestehende Datenmodell, beispielsweise numerische Prozentzahlen eines Wahlergebnisses als Tabelle, als Balken- oder Kreisdiagramm. Hierdurch ergibt sich eine nahezu beliebige Skalierbarkeit des Systems.

4.3 Projektmanagement

Das folgende Kapitel befasst sich mit allen organisatorischen und planungstechnischen Schritten, die zu einem erfolgreichen Projektmanagement zählen. Hierzu gehören wie aus Kapitel 4.2.3 bekannt die Planung des Projektteams ebenso wie die Zeitplanung und ein Lastenheft.

4.3.1 Projektname „Zwickr“

Vor dem Einstieg in die Projektarbeit, ob bei kleineren oder sehr großen Projekten, sollte ein Projektname definiert werden. Dieser Titel sorgt innerhalb des Teams für eine unmissverständliche Kommunikation und verleiht dem Projekt eine gewisse Persönlichkeit. Nicht zu verwechseln ist dieser Name mit dem späteren Produktnamen. In seltenen Fällen können diese Namen zwar übereinstimmen, wenn beispielsweise die Bezeichnung des Produkts schon vor der Entwicklung hundertprozentig feststeht. In der Regel entsteht so eine Produktnamensfindung allerdings mit der Zeit der Entwicklung und nach einigen Marketingtests.

Die in dieser Diplomarbeit entstandene Anwendung trägt den Codenamen *Zwickr*. Dieser Name wurde nach einigen projektvorbereitenden Meetings besitmmt und steht für die bis zu Beginn des 20. Jahrhunderts gebräuchliche Sehhilfe Zwicker, eine Brille ohne Bügel. Zwickr soll Einblick in die Welt der Daten bieten und Durchblick verschaffen, also eine Sehhilfe sein. Durch Zwickr kann der Blick auf Datenmengen gezielt gerichtet und an bestimmter Position, im Sinne des Fokussierens, geschärft werden.

4.3.2 Team

Für die Entstehung einer Webanwendung ist nur in seltenen Fällen eine Person alleine verantwortlich. Je nach Projektgröße findet sich ein Team von Entwicklern, Designern und Konzeptionern zusammen. Die Projektleitung, welche über die Anwendungsanforderungen informiert

ist, entscheidet, wie viele Entwickler, Designer und Konzeptioner benötigt werden. Das Team besteht entweder aus einer Reihe interner Mitarbeiter oder wird zusätzlich mit externen Mitarbeitern ergänzt. Hierbei spielen verschiedene Aspekte eine Rolle. Bieten die eigenen Mitarbeiter das Know-how? Haben die eigenen Mitarbeiter noch freie Kapazitäten für ein weiteres Projekt? Für welche Aufgaben benötigt das Team externe Unterstützung? Wie setzt sich die Kostenkalkulation zusammen?

Für die Entwicklung des Zwickr-Prototyps wurde ein Team aus drei Personen zusammengestellt. Dabei wurden die Aufgaben grob in die Bereiche Projektleitung, Entwicklung und Design unterteilt. Die Autorin dieser Diplomarbeit, Janina Werner, übernahm hierbei in erster Linie die Projektleitung, wirkte aber ebenfalls in der Gestaltung und Entwicklung mit. Obwohl eine grobe Rollenverteilung vorlag, fand stets ein reger Austausch auf kommunikativer wie auch auf technischer Ebene statt. So wusste jeder genauestens über die Arbeit der Kollegen Bescheid und konnte teilbereichsübergreifend höchst effizient an dem Projekt mitwirken. In welchem zeitlichen Rahmen die einzelnen Aufgaben von wem bearbeitet wurden, ist dem Projektzeitplan [vgl. Anlage A1] zu entnehmen.

4.3.3 Zeitmanagement

Wie schon in Kapitel 4.2.3 beschrieben, ist ein gut kalkulierter Zeitplan grundlegender Bestandteil im Rahmen der Softwareentwicklung. Die zuvor definierten Meilensteine werden in einen zeitlichen Zusammenhang gebracht und mit den einzelnen Projektmitarbeitern abgestimmt. Dabei erscheint es als sehr sinnvoll, zunächst jedes Teammitglied seine Aufgaben selbst kalkulieren zu lassen, da niemand sonst so viele spezifische Erfahrungswerte vorzuweisen hat.

Zur Zeitplanung der vorliegenden Arbeit wurde ein sogenanntes *Gantt-Diagramm*, auch Balkenplan genannt, verwendet. Hierin wird die zeitliche Abfolge von Aktivitäten grafisch in Form von waagerechten Balken

auf einer Zeitachse dargestellt. Je länger ein Balken ist, desto länger dauert die Aktivität. Überschneiden sich zeitlich zwei Aktivitäten, so werden auch die Balken überlappend dargestellt.

Der Vorteil zu anderen Zeitplanmodellen, wie beispielsweise einem Netzplan besteht darin, dass eine Aktivitätsdauer auf Anhieb sichtbar ist. Es sollte jedoch darauf geachtet werden, dass viele Einzelaktivitäten der Übersicht zuliebe zu Projektphasen zusammengefasst werden, so dass eine gute Übersicht gewährleistet bleibt. [W09g]

Der Zeitplan zum vorliegenden Projekt befindet sich in den Anlagen unter dem Gliederungspunkt A1.

4.3.4 Lastenheft

Als grobe Produktskizze der zu erstellenden Software-Applikation gilt das sogenannte *Lastenheft*. Es dient der Beschreibung aller fachlichen Anforderungen (Leistungs- und Lieferumfang) an die fertige Software bzw. der Anwendung, aus Sicht des Auftraggebers. Mithilfe dieses ersten Dokuments im gesamten Entwicklungsprozess wird dem Programmierer erklärt, was entwickelt werden soll. Zusätzlich wird jeder einzelne Arbeitsschritt mit einer laufenden Nummer zugehörig zu Produktfunktionen („LF“ für Produktfunktionen des Lastenheftes), Produktdaten („LD“ für Produktdaten des Lastenheftes) und Produktleistungen („LL“ für Produktleistungen des Lastenheftes) versehen, so dass eine einheitliche Sprache bei der Kommunikation (schriftlich wie verbal) gewährleistet ist [B09].

Das Lastenheft zum vorliegenden Projekt befindet sich in den Anlagen unter dem Gliederungspunkt A2.

5 Das RIA-Framework Flex 3

Adobe Flex, das Framework zur Erstellung zukunftsorientierter Enterprise-Webanwendungen, gibt es bereits seit März 2004 – damals noch unter Macromedia mit der Version 1.0 publiziert. Das nächste Major-Release, Flex 2.0, wurde im Juni 2006, kurz nach der Übernahme von Macromedia durch Adobe Systems, veröffentlicht. Obwohl in Flex erstellte Anwendungen, also `*.swf`- oder `*.air`-Dateien, auf allen gängigen Betriebssystemen (Microsoft Windows, Mac OS X und Linux) oder als Desktopanwendungen laufen, zeigen die meisten Webentwickler erst seit der Markteinführung der aktuellen Version, Flex 3, die seit Februar 2008 auf dem Markt ist, wirkliches Interesse an dem Produkt. Dies liegt in verschiedenen Begebenheiten begründet.

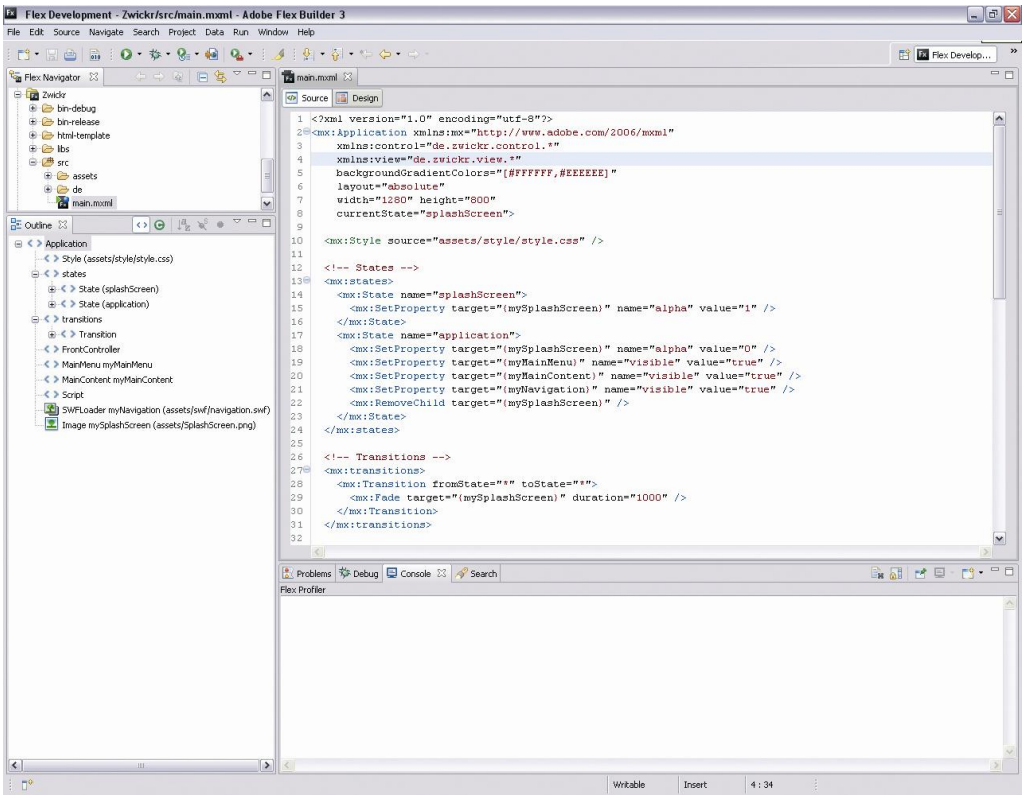
Zum einen hängen die gestiegenen Verkaufszahlen vermutlich mit der inzwischen allgemein größeren Akzeptanz und Verbreitung des Flash Players zusammen [vgl. Tab. 1], zum anderen sicherlich auch mit dem veränderten und daher erstmals auch für klein- und mittelständische Unternehmen wesentlich attraktiveren Lizenzmodell von Adobe [W08b]:
Lediglich für den Flex Builder 3 fallen Kosten an, die für die Standard-Variante 213 € und für die Professional-Variante 594 € betragen [AOS09]. Das Flex SDK (Software Development Kit) ist open-source unter der Mozilla Public License verfügbar.

5.1 Die Bestandteile von Flex 3

Neben dem kostenlosen Flex SDK bietet Flex 3 sowohl die auf Eclipse basierende proprietäre IDE Flex Builder, bestehend aus einem Compiler, einem Debugger und einem visuellen CSS-Editor, als auch die LiveCycle Data Services (ehemals als Flex Data Services bekannt). Diese stehen dem Entwickler für die Nutzung innerhalb einer CPU und einer Applikation open-source zur Verfügung und bieten beispielsweise über

die kostenlose Schnittstelle BlazeDS Möglichkeiten, Serveranwendungen anzusprechen.

In Form einer Erweiterung des Flex SDK (nicht im Flex SDK enthalten) stellen die sogenannten *Charting Components* dem Entwickler auf optisch ansprechende Weise interaktive Diagramme zur Verfügung. [Wo8b]



20 - Entwicklungsumgebung Flex Builder 3

5.2 Programmieren in Flex

Die Entwicklungssprachen in Flex sind zum einen MXML und zum anderen das aus Flash CS4 bekannte ActionScript 3.0. Mit der deklarativen Layoutsprache MXML, die auf XML basiert, kann das gesamte Erscheinungsbild der Anwendung durch sichtbare und unsichtbare Komponenten definiert werden. Dazu zählen neben Layoutkomponenten wie Textfelder, Schaltflächen und Bilder auch sogenannte *View States* und *Transitions*, die die Reaktion der Anwendung auf bestimmte

Benutzeraktionen, bzw. die Übergänge zwischen zwei Zuständen definieren. [Wo8b]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:Panel title="Beispiel für eine MXML-Struktur">
    <mx:Button id="myButton" label="Dies ist ein Button" />
  </mx:Panel>
</mx:Application>
```

Quellcode 1 - Beispiel für eine MXML-Struktur

Die eigentliche Programmlogik beschreibt die leistungsfähige und objektorientierte Programmiersprache ActionScript 3.0. Sie beruht auf dem internationalen Standard ECMAScript, besser bekannt als JavaScript. [Wo8b]

```
public function ausgabeFunktion (
  vorname:String,
  nachname:String,
  alter:Number ) : String {
  return vorname+" "+nachname+" ist "+alter+" Jahre alt ";
}
public function main() : void {
  var vorname:String = "Janina";
  var nachname:String = "Werner";
  var alter:Number = 27;
  var  ausgabe:String  =  ausgabeFunktion(  vorname,  nachname,
    alter );
  trace( ausgabe );
  //Gibt in der Console aus: "Janina Werner ist 27 Jahre alt"
}
```

Quellcode 2 - Beispiel für zwei ActionScript-Funktionen

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">

  <mx:Script>
    <![CDATA[
      public function ausgabeFunktion (
        vorname:String,
        nachname:String,
        alter:Number ) : String {
          return  vorname+"  "+nachname+"  ist  "+alter+"  Jahre
            alt ";
      }
      public function main() : void {
        var vorname:String = "Janina";
        var nachname:String = "Werner";
        var alter:Number = 27;
        var ausgabe:String = ausgabeFunktion( vorname,
          nachname, alter );
        trace( ausgabe );
        //Gibt in der Console aus: "Janina Werner ist 27
          Jahre alt"
      }
    ]]>
  </mx:Script>

  <mx:Panel title="Beispiel für eine MXML-Struktur">
    <mx:Button id="myButton" label="Dies ist ein Button" />
  </mx:Panel>
</mx:Application>
```

Quellcode 3 - Beispiel für in MXML eingebetteten ActionScript-Code

In einem Zwischenschritt dient der Flex-Compiler der „Übersetzung“ von MXML-Dateien in ActionScript-Quelldateien, die dann wiederum zu einer Flashdatei (`*.swf`) kompiliert werden. Zum Ausführen der Datei ist beim Benutzer der installierte Flash Player ab Version 9 notwendig. [Wo8b]

5.3 Möglichkeiten der Datenanbindung

Im Hinblick auf die Zielsetzung des Projekts, nämlich die Entwicklung einer Rich Internet Application mit Datenbankanbindung, wird an dieser Stelle auf die verschiedenen Möglichkeiten der Anbindung externer Daten an Flex eingegangen, die Simon Widjaja wie folgt schildert [Wo8c]:

Zunächst sei bekannt, dass für das Flex-Framework keine direkte Verbindung zu einer Datenbank vorgesehen ist. Dies bedeutet, dass alle Daten, die zum Kompilierungszeitpunkt nicht vorliegen, zur Laufzeit von einem Server-Backend in die Applikation nachgeladen werden müssen. Zur Erstellung dieses Backends können beispielsweise Java, PHP, .NET oder Coldfusion genutzt werden, aber auch jede andere Technologie, die beispielsweise XML-Datenströme über HTTP ermöglicht. Grundsätzlich werden zwei Arten der Kommunikation unterschieden: Es gibt zum einen die HTTP-basierte Kommunikation und zum anderen die Echtzeitkommunikation, die vor allem bei der Entwicklung von Computerspielen eine Rolle spielt. Im Flex-Framework ist jedoch nur die HTTP-basierte Kommunikation vorgesehen, daher wird diese in der vorliegenden Arbeit behandelt.

Die HTTP-basierte Kommunikation funktioniert nach dem Anfrage-Antwort-Prinzip. Sobald eine Anfrage vom Client ausgeht, wird eine Verbindung zum Server aufgebaut und diese wieder geschlossen, sobald die Antwort übermittelt wurde. Dabei können Texte, URL-kodierte Daten, XML- oder SOAP-Datenströme übertragen werden. Für die HTTP-basierte Kommunikation sind in Flex verschiedene Klassen mit serverseitigen Datenquellen, wie die des Statistischen Bundesamtes Deutschland, vorgesehen. Diese Klassen gehören zu den sogenannten *Remote Procedure Calls* (kurz RPC), sie lauten `HTTPService`, `WebService` und `RemoteObject`.

5.3.1 HTTPService

Der Versand von Daten über den `HTTPService` funktioniert via GET und POST sowohl über HTTP als auch über HTTPS. Üblicherweise wird diese Klasse für den Transfer von XML-Daten verwendet. Über einen `HTTPService`-Tag in MXML wird die URL der zu ladenden Datei und eine eindeutige ID definiert. [Wo8d]

```
<mx:HTTPService id="myService" url="daten/janina.xml />
```

Quellcode 4 - Beispiel für einen HTTPService-Tag

Eine `send()`-Methode sorgt dann für die Ausführung des Datentransfers. Dies kann entweder nach einer Benutzerinteraktion, beispielsweise über einen Button, erfolgen, oder aber nach einem Systemereignis. Es ist darauf zu achten, dass auf zwei Events der `HTTPService`-Klasse reagiert wird: Bei erfolgreichem Dateneingang wird das `result`-Event ausgelöst, so dass auf die eingetroffenen Daten zugegriffen werden kann. Kam es zu einer fehlerhaften Übertragung, wird das `fault`-Event gefeuert. An dieser Stelle könnte beispielsweise eine Fehlermeldung den Benutzer per Alert-Fenster darauf hinweisen, dass die Datenübertragung nicht vollständig stattgefunden hat. [Wo8d]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="vertical"
  creationComplete="init(event)">

  <mx:Script>
    <![CDATA[
      import mx.utils.ObjectUtil;
      import mx.rpc.events.FaultEvent;
      import mx.rpc.events.ResultEvent;
      import mx.controls.Alert;
      private function init( event:Event ):void {
        myService.send();
      }
      private function onResult( event:ResultEvent ):void {
        Alert.show( "Daten sind eingegangen.", "RESULT" );
      }
      private function onFault( event:FaultEvent ):void {
        Alert.show( event.fault.faultDetail,
          event.fault.faultString );
      }
    ]]>
  </mx:Script>
```

```
  <mx:HTTPService id="myService" url="daten/janina.xml"
    result="onResult(event)" fault="onFault(event)" />
  <mx:DataGrid
    dataProvider="{myService.lastResult.root.product}" />

</mx:Application>
```

Quellcode 5 - Beispiel für die Verwendung des HTTPService

5.3.2 WebService

Der `WebService` stellt laut W3C eine Schnittstelle für die Kommunikation zwischen zwei Maschinen dar. In diesem Fall ist eine Datenübertragung im SOAP-Format üblich. Der Aufruf dieses Services verhält sich ähnlich dem zuvor erläuterten `HTTPService`. [W08e]

5.3.3 RemoteObject

Die beiden zuvor vorgestellten Klassen zur Datenübertragung eignen sich in hohem Maß für den Transfer von relativ kleinen, bzw. weniger komplexen Datenmengen. In dem vorliegenden Fall der Anbindung einer Rich Internet Application an Massendaten liegt der Fokus auf der dritten Möglichkeit der Datenübertragung. Das sogenannte `RemoteObject` ermöglicht über das *Action Message Format* (kurz AMF) den Versand eines binären Datenstroms direkt von einer serverseitigen Quelle an Flex und umgekehrt. Daten werden also über eine Backend-Schnittstelle vor dem Versand in das AMF-Format serialisiert und anschließend clientseitig deserialisiert. Sie liegen demnach beim Empfänger wieder in der ursprünglichen Form vor, wobei inkompatible Datentypen entsprechend einer Mapping-Tabelle dem Zielformat angepasst werden. Um das `RemoteObject` nutzen zu können, müssen zugehörige Konfigurationsdaten dem Compiler über Hinzufügen einer Konfigurationsdatei in den Projekteinstellungen bekannt gemacht werden. [W08f]

Abb. 21 - Ausschnitt des Compiler-Menüs

5.4 Mikroarchitektur-Framework Cairngorm

Cairngorm ist ein speziell für die Entwicklung mit dem Flex-Framework entwickeltes Mikroarchitektur-Framework, das Adobe Systems als Open-Source-Produkt zum Download anbietet:
`http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm`.

Ein Mikroarchitektur-Framework ist ein Werkzeug bzw. Regelwerk, welches in erster Linie bei der strikten Umsetzung des Prinzips der lose gekoppelten Programmierung [vgl. Kap. 4.2.3.2] hilft. Gerade bei der Enterprise-Entwicklung dient so ein zusätzliches Framework mithilfe ausgewählter Entwurfsmuster der Realisierung einer sauberen und strukturierten Anwendung, die zudem wartungsfreundlich und gut erweiterbar ist. In der Regel wachsen selbst weniger umfangreich konzipierte Projekte schnell über den ursprünglichen Rahmen hinaus, so dass Erweiterungen integriert und Strukturen verändert werden müssen. Deshalb wird für die Entwicklung der vorliegenden Anwendung auf ein solches Framework gesetzt.

Zu den im Mikroarchitektur-Framework Cairngorm verankerten Design-Pattern gehören:

- Front Controller
- Command Pattern
- Business Delegate
- Service Locator
- Value Objects (VO), bzw. Data Transfer Objects (DTO)

Das folgende Diagramm veranschaulicht den Aufbau einer typischen Cairngorm-Architektur:

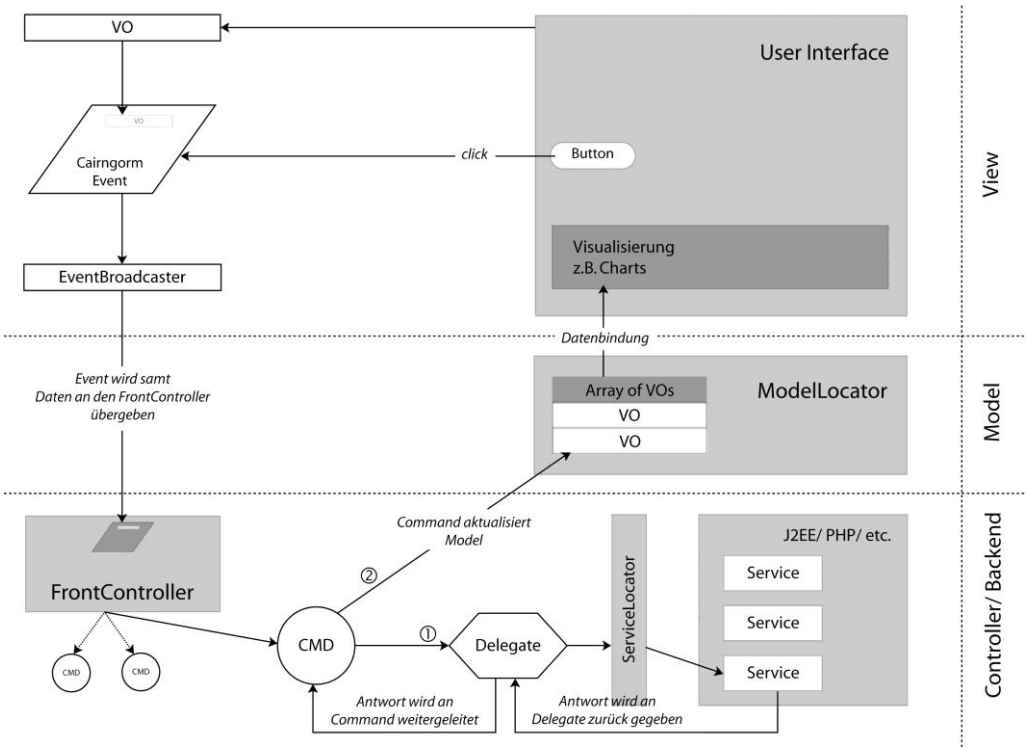


Abb. 22 - Cairngorm-Architektur

Zusammenfassend kann die Arbeitsweise von Cairngorm allgemein folgendermaßen beschrieben werden:
Die View feuert (engl. to dispatch) Events, die vom Controller aufgefangen werden, der wiederum dementsprechende Commands instanziiert. Diese Commands sind von der View entkoppelt und verändern nur die Daten im Model. Das Model wiederum aktualisiert über Datenbindung (engl. Data Binding) die View-Komponenten.

Des Weiteren dient die Klassenbibliothek `cairngorm.swc` als Grundlage für Klassenerweiterungen im Projekt [vgl. Quellcode 7].

6 Hauptapplikation Zwickr

In diesem Kapitel werden alle Aspekte der technischen Entwicklung von Zwickr auf Basis von Flex 3 behandelt und diskutiert. Dazu gehören Teilbereiche wie die Projektarchitektur, das Datenmodell, die Schnittstellen und die Interaktionen zwischen den einzelnen Modulen.

6.1 Projektarchitektur

Zu Beginn dieses Projektes, mit dem ersten Schritt des Prototyping, steht noch nicht genau fest, welchen Umfang die Projektstruktur insgesamt annehmen könnte. Wie aus Kapitel 5.4 bekannt ist, sollte aber bereits bei überschaubaren Projekten ein Mikroarchitektur-Framework eingesetzt werden, um das Projekt agil zu halten. Aus diesem Grunde baut auch die vorliegende Anwendung auf dem Design-Pattern-Modell des Architektur-Frameworks Cairngorm auf.

Innerhalb dieser Struktur befinden sich Ordner, die beim Anlegen eines neuen Flex-Projektes automatisch erstellt werden. Dazu gehören `bin-debug`, `bin-release`, `html-template`, `libs` und `src`.

Wie aus Abb. 24 entnommen werden kann, enthält das Projekt Zwickr jedoch keinen Ordner `html-template`. Dies liegt an einer in Entwicklerkreisen üblichen Voreinstellung des Projekts, die zum Kompilieren der Anwendung kein *HTML wrapper file* generieren lässt. Damit wird erreicht, dass die Anwendung im sogenannten *Standalone Flash Player* ausgeführt wird.

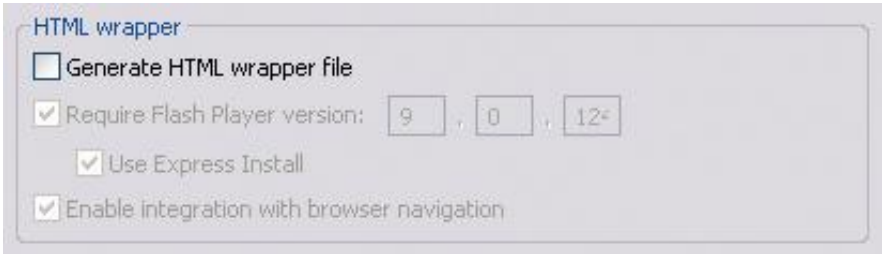


Abb. 23 - Projekteinstellungen zum HTML wrapper

Die folgende Abbildung zeigt die Ordnerstruktur des Projekts Zwickr:

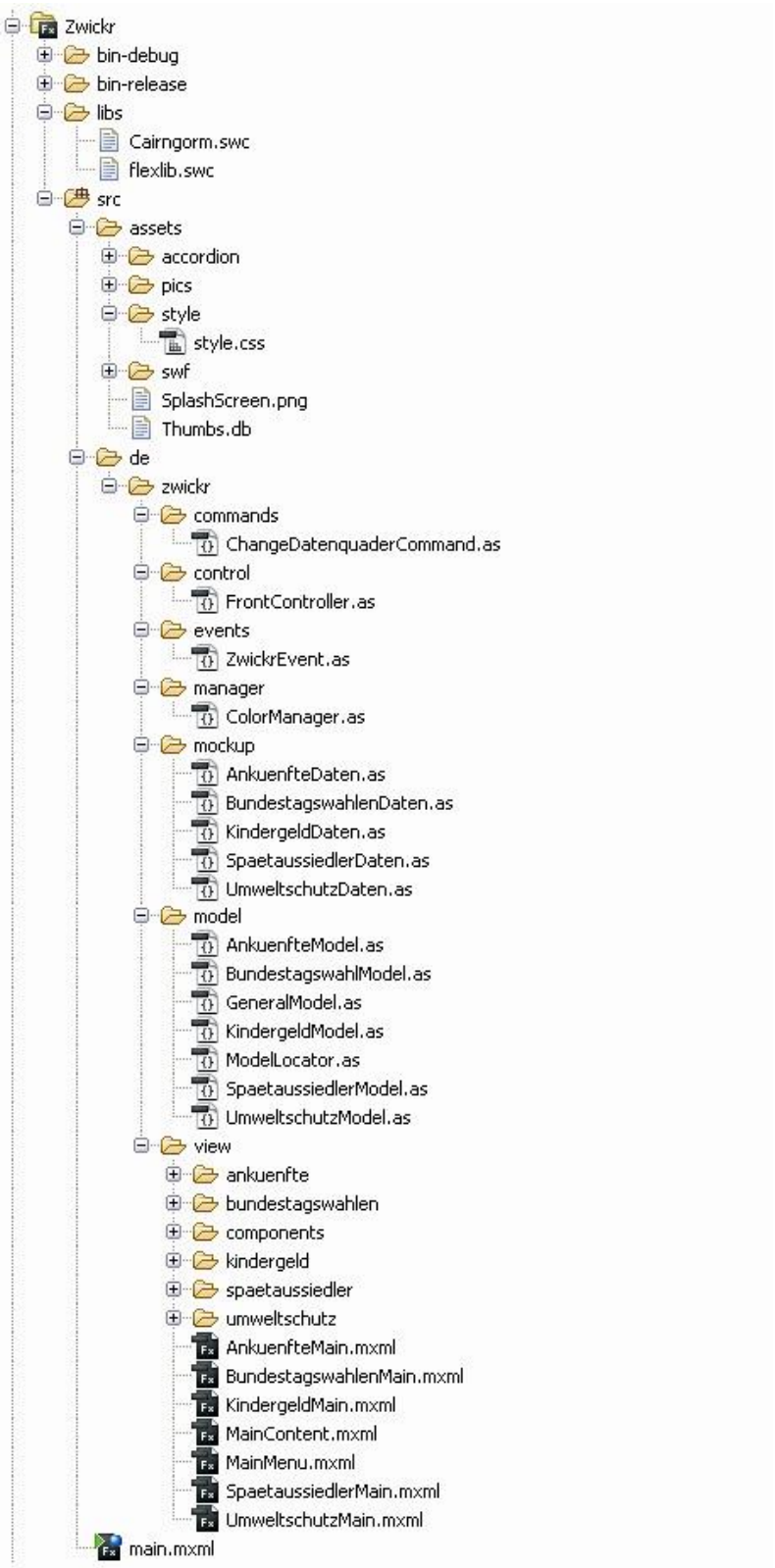


Abb. 24 - Ordnerstruktur des Projekts Zwickr

Im `libs`-Ordner können externe Klassen-Bibliotheken abgelegt werden, so auch im vorliegenden Projekt die Bibliothek `cairngorm.swc`. Eine externe Bibliothek wird durch Kopieren in den `libs`-Ordner dem Projekt hinzugefügt und kann damit bereits verwendet werden, da der `libs`-Ordner in den Projekteinstellungen als Ort für externe Bibliotheken eingetragen ist.
Die eigentliche Anwendung untergliedert sich im vorliegenden Projekt in weitere sieben Unterordner, die nach dem Prinzip des Best Practice Management [vgl. Kap. 4.2.3] innerhalb des `src`-Ordners unter dem Pfad `src/de/zwickr` abgelegt werden. Auf diese Ordnerstruktur und die Inhalte der einzelnen Ordner wird in den folgenden Kapiteln genauer eingegangen.

6.1.1 Main-Applikation

Der Einstieg in die Hauptapplikation ist die Main-Applikation `main.mxml`. Sie befindet sich in der Ordnerhierarchie des Projekts auf oberster Ebene und ist der Startpunkt der Anwendung. Die Main-Applikation sorgt für die Instanziierung aller weiteren Komponenten. Die folgende Abbildung veranschaulicht den Aufbau der `main.mxml`.

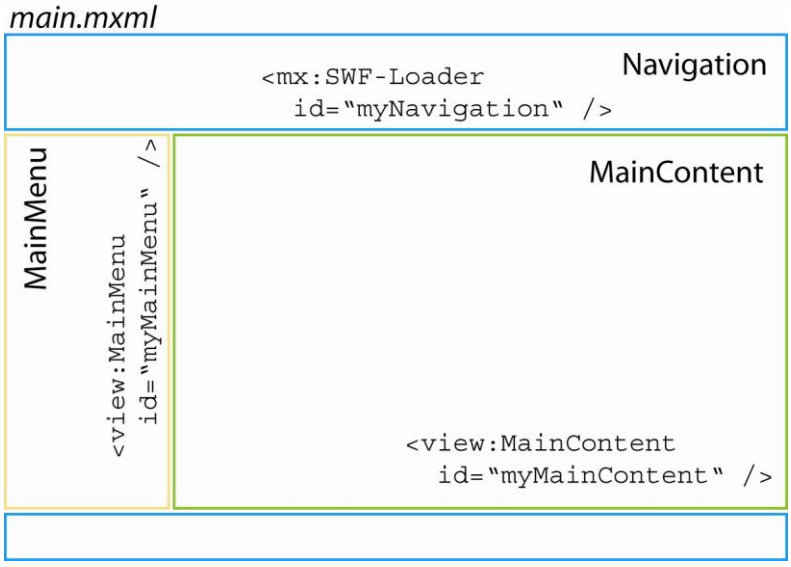


Abb. 25 - Aufbau der main.mxml

Die Hauptapplikation setzt sich aus drei Bereichen zusammen: der View `MainMenu` (gelb), der View `MainContent` (grün) und der Flash-Navigation `navigation.swf` (blau). Anhand ihrer Instanziierungs-Tags ist zu erkennen, dass verschiedene *Namespaces* verwendet werden. Namespaces dienen u.a. der Nutzung eigener Komponenten, ohne das Risiko einzugehen, auf Konflikte durch nicht eindeutige Namensgebung zu stoßen. Namespaces werden im `Application`-Tag mit `xmlns` angegeben, für die `main.mxml` bedeutet dies:

```
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:control="de.zwickr.control.*"
    xmlns:view="de.zwickr.view.*" >

. . .

</ mx:Application>
```

Quellcode 6 - Namespaces der Hauptapplikation

Der Standard-Namespace `mx` ist dem Universal Resource Identifier (kurz URI) `http://www.adobe.com/2006/mxml` zugewiesen, worüber die Standard-Flex-Komponenten bereit gestellt werden. Die Namespaces `control` und `view` dienen der Pfadangabe zu den jeweiligen Verzeichnissen innerhalb der Projektstruktur [vgl. Quellcode 6]. Dies entspricht dem voll qualifizierten Pfad des Packages in ActionScript.

Daraus ergeben sich die Instanziierungs-Tags `<view: />`, `<mx: />` und `<control: />`. Der `control`-Tag instanziiert die `FrontController`-Klasse, die in der Ordnerstruktur des Projekts unter dem Pfad `src/de/zwickr/control/FrontController.as` abgelegt ist.

```
package de.zwickr.control
{
  import com.adobe.cairngorm.control.FrontController;

  import de.zwickr.commands.ChangeDatenquaderCommand;
  import de.zwickr.events.ZwickrEvent;

  public class FrontController extends
    com.adobe.cairngorm.control.FrontController
    {
```

```
        public function FrontController() {
          super();
          setupCommands();
        }

        private function setupCommands():void {
          addCommand( ZwickrEvent.CHANGE_DATENQUADER,
            ChangeDatenquaderCommand );
        }
    }
}
```

Quellcode 7 - FrontController.as

Die `FrontController`-Klasse erweitert die Superklasse `Front-Controller` der Cairngorm-Klassenbibliothek, wodurch diese für die speziellen Anforderungen an das Projekt Zwickr angepasst werden kann. Der `FrontController` nimmt alle Cairngorm-Events entgegen und delegiert diese an die entsprechenden Commands.

6.1.2 MainMenu

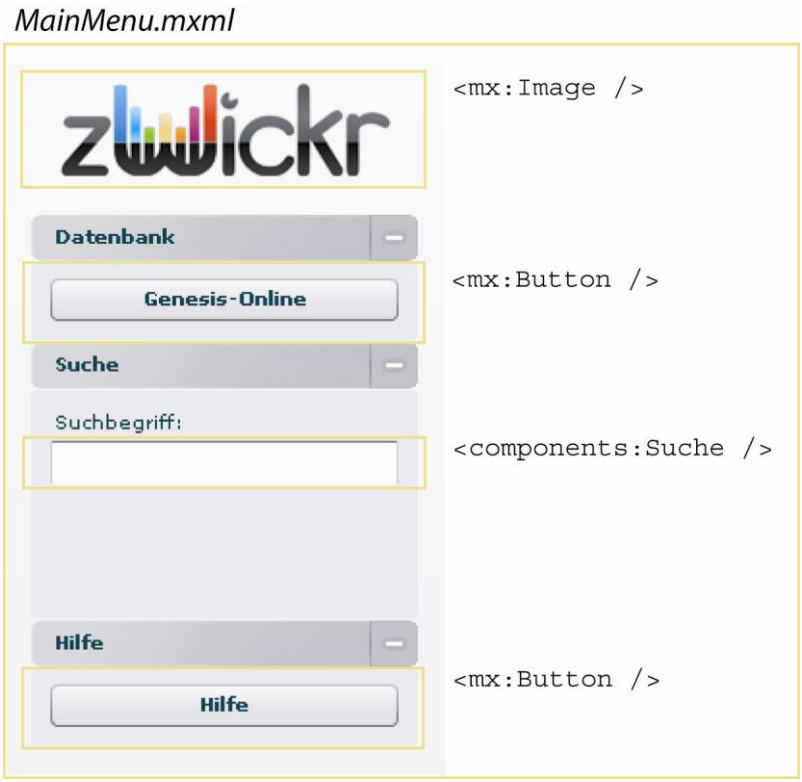


Abb. 26 - Aufbau der MainMenu.mxml

Die View `MainMenu` ist eine `VBox`, also ein Layout-Container, in der alle Kindelemente vertikal untereinander angeordnet werden. Sie beinhaltet das Logo des Projekts Zwickr und drei Instanzen der *Custom Component*, also einer von Grund auf neu entwickelten Komponente, dem `CollapsableMenuItem`. Diese Komponente kann jedes beliebige `DisplayObject` beinhalten. Eine Komponente neu zu entwickeln macht dann Sinn, wenn eine Standardkomponente die Anforderungen an sie nicht erfüllt. So erlaubt es das `CollapsableMenuItem`, innerhalb des `MainMenu` mehrere Menüelemente parallel geöffnet zu haben, welche Eigenschaft der Navigator-Container `Accordion` beispielsweise nicht bietet.

Die Instanziierung jeder einzelnen Custom Component erfolgt über den entsprechenden Namespace:

```
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:components="de.zwickr.view.components.*"
    verticalGap="1">

. . .

  <components:CollapsableMenuItem label="Suche">
    <components:content>
      <mx:Panel width="100%" styleName="accordionPanel">
        <components:Suche />
      </mx:Panel>
    </components:content>
  </components:CollapsableMenuItem>

. . .

</ mx:VBox>
```

Quellcode 8 - Ausschnitt der MainMenu.mxml

Das `MainMenu` setzt sich aus zwei Standardkomponenten (Button) und wiederum einer Custom Component `Suche` zusammen:

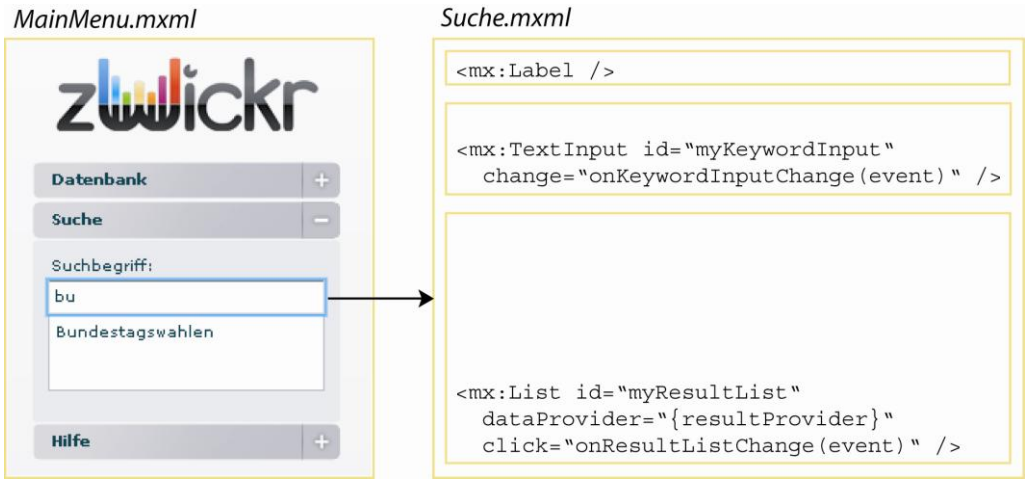


Abb. 27 - Aufbau der Suche.mxml

Wie Abb. 27 zu entnehmen ist, besteht die Custom Component `Suche` aus einem Label, einem Texteingabefeld und einer Liste. Findet über das Eingabefeld eine Benutzerinteraktion (engl. User Gesture) durch Eingabe eines oder mehrerer Buchstaben statt, wird das `change`-Event gefeuert und der Event Handler `onKeywordInputChange()` aufgerufen. Dieses übergibt den eingegebenen Text als Variable `keyword` vom Typ `String` an die Methode `search()`.

```
private function onKeywordInputChange( event:Event ):void {
  var keyword:String = myKeywordInput.text;
  search( keyword );
}
```

Quellcode 9 - Event Handler onKeywordInputChange()

Innerhalb dieser Methode findet ein Vergleich der `keyword`-Variable mit den im `dataProvider` eingetragenen Metatags statt. Sobald der eingegebene String aus mindestens einem Zeichen bestehend zu einer Übereinstimmung führt, werden alle passenden Suchbegriffe in der Liste angezeigt. Der Benutzer kann nun über die Listeneinträge zu der View des jeweiligen Datenquaders navigieren. Dies geschieht über das `click`-Event, welches durch die Benutzerinteraktion gefeuert wird. Die `onResultListChange()`-Methode fängt das `selectedItem`, also den

ausgewählten Listeneintrag, ab und feuert ein neues `ZwickrEvent` mit dem aktuell ausgewählten Datensatz.

6.1.3 MainContent

Die View `MainContent` wird durch den Layout-Container `Canvas` beschrieben. In ihr werden fünf States definiert, die jeweils eine der folgenden Content-Views beinhalten: `SpaetaussiedlerMain`, `KindergeldMain`, `AnkuenfteMain`, `UmweltschutzMain` oder `BundestagswahlenMain`. Welche der Views gerade angezeigt wird, bestimmt die Eigenschaft `currentMainState` des `GeneralModels`, zu dem eine Datenbindung besteht, zu erkennen an den geschweiften Klammern („`{}`“).

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:view="de.zwickr.view.*"
    currentState="{ model.generalModel.currentMainState }">

. . .

</ mx:Canvas>
```

Quellcode 10 - Datenbindung an das GeneralModel

Die Funktionsweise von States ist in etwa mit mehreren visuellen Ebenen zu vergleichen, die abwechselnd sichtbar werden [vgl. Abb. 28].

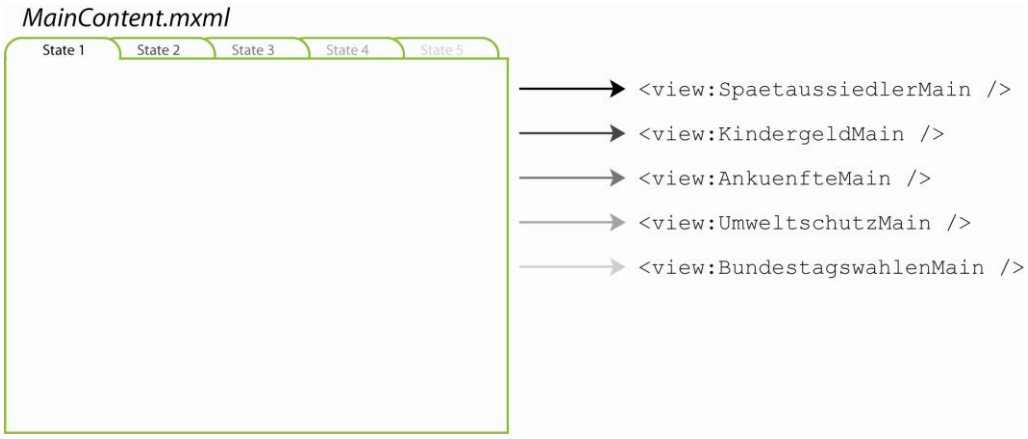


Abb. 28 - Aufbau der MainContent.mxml

6.1.4 Views

Wie in Kapitel 6.1.3 beschrieben, wird im `MainContent` einer von fünf verschiedenen States angezeigt. Die Views, die sich innerhalb dieser States befinden, sind ähnlich aufgebaut. Sie beinhalten wiederum zwei bis drei States, für jede Diagramm-View einen und das dazugehörige Menü, welches wiederum aus mehreren Instanzen der bereits bekannten Custom Component `CollapsableMenuItem` besteht [vgl. Kap. 6.1.2]. Zu jeder View existiert ein dazugehöriges Model, in dem die jeweiligen Daten der Diagramme (Charting Components) gehalten werden. Zu jedem Menü einer View existiert wiederum ein `MenuController`, der die Benutzerinteraktionen mit dem Menü abfängt und auswertet. Mithilfe der Datenbindung führt eine Änderung im Menü durch eine Benutzerinteraktion zur sofortigen Anpassung der zugehörigen View. Die Kommunikation zwischen Menü und Diagramm-View übernimmt dabei ebenfalls der `MenuController`.

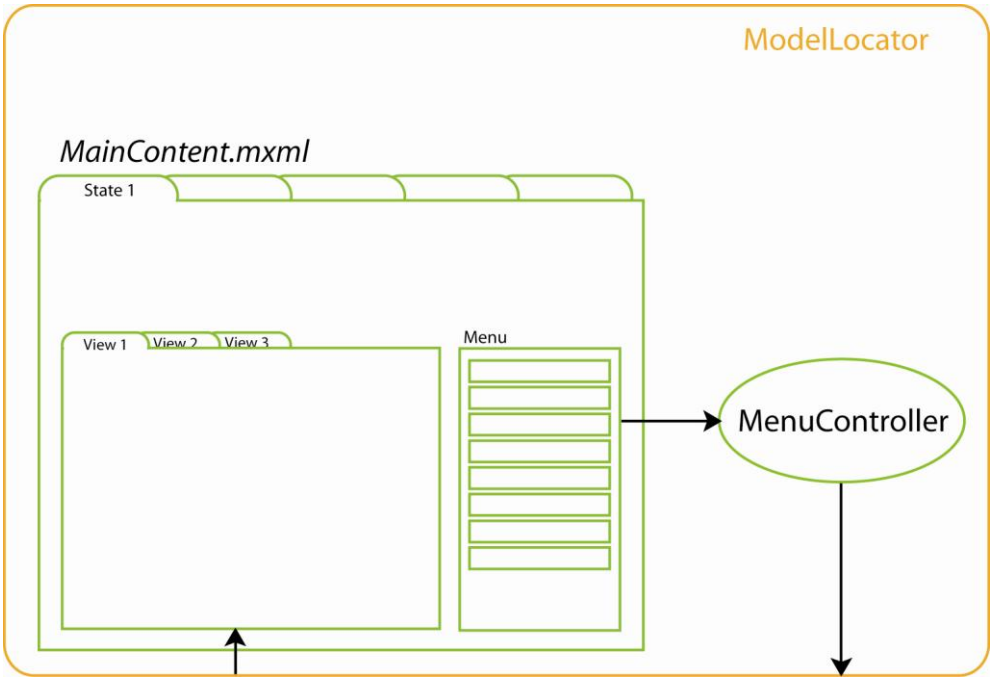


Abb. 29 - Kommunikation zwischen View und Menü

Da der `ModelLocator` eine global verfügbare Klasse sein soll und ActionScript3 keine Möglichkeit bietet, den Konstruktor `private` zu setzen, ist anderweitig sicherzustellen, dass von dieser Klasse kein weiteres Objekt erzeugt werden kann. Zu diesem Zweck existiert das Design Pattern `Singleton`, das im `ModelLocator` durch die interne Klasse `SingeltonEnforcer` gewährleistet, dass der Konstruktor nur innerhalb der `Singleton`-Klasse aufgerufen werden kann.

```
package de.zwickr.model
{
  import com.adobe.cairngorm.model.IModelLocator;

  [Bindable]
  public class ModelLocator implements IModelLocator
  {

. . .

    private static var _instance:ModelLocator;

    public static function getInstance():ModelLocator {
      if ( !_instance ) {
        _instance = new ModelLocator( new SingletonEnforcer()
        );
      }
      return _instance;
    }

    public function ModelLocator(
      singletonEnforcer:SingletonEnforcer ) {
    }

  }
}
class SingletonEnforcer {}
```

Quellcode 11 - Singleton

6.1.5 Navigation

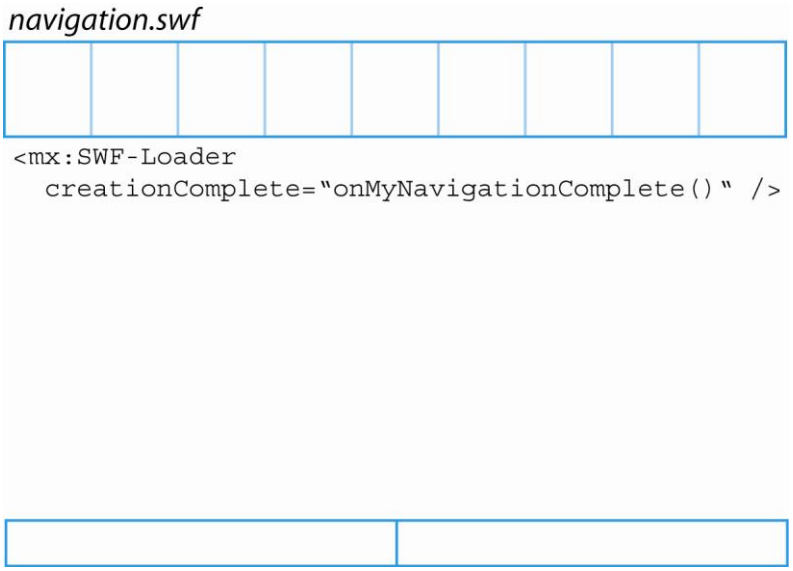


Abb. 30 - Aufbau der Navigation.swf

```
<mx:SWFLoader id="myNavigation"
    source="assets/swf/navigation.swf"
    creationComplete="onMyNavigationComplete()" />
```

Quellcode 12 - SWFLoader

Die Navigation der Anwendung setzt sich aus zwei Hauptteilen zusammen: der eigentlichen Navigation am oberen Rand der Anwendung und der sogenannten Sammlung am unteren Rand der Anwendung. Diese teilt sich wiederum in zwei Bereiche auf: einmal die Ablage für recherchierte Themen und einmal die Ablage für Infografiken, die temporär gespeichert werden sollen. Da es sich bei der Navigation um einen Flash-Film handelt, wird dieser als `*.swf`-Datei in die Zwickr-Anwendung geladen. Hierfür wird die Flex-Komponente `SWFLoader` verwendet. Sobald die Navigation vollständig erstellt ist (`creationComplete`), wird der Event Handler `onMyNavigationComplete()` aktiv.

```
private function onMyNavigationComplete():void {
  myNavigation.addEventListener( 'navigationClick',
    onNavigationClick );

. . .

}
```

Quellcode 13 - Methode onMyNavigationComplete()

Der Event Handler ist dafür zuständig, das gefeuerte Flash-Event mit dem Event-Namen `navigationClick` abzufangen, um dann wiederum einen weiteren Event Handler für das `navigationClick`-Event zu registrieren. Dies wird durch die Referenz auf die Methode `onNavigationClick()` als zweiten Paramater der Methode `addEventListener()` realisiert.

```
private function onNavigationClick( event:DataEvent ):void {
  new ZwickrEvent( ZwickrEvent.CHANGE_DATENQUADER,
    event.data ).dispatch();
}
```

Quellcode 14 - Event Handler onNavigationClick()

Der Event Handler `onNavigationClick()` bekommt den Parameter `event` vom Typ `DataEvent` übergeben. Wie die in Flex üblich verwendeten Events stammt auch dieses vom `FlexEvent` und damit wiederum vom `Event` ab, wie die folgende Vererbungskette zeigt:

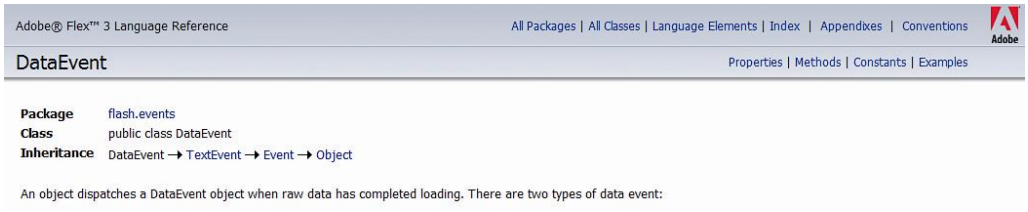


Abb. 31 - Vererbungskette des DataEvent

Zunächst instanziiert dieser Event Handler nun ein `ZwickrEvent`. Die Besonderheit an diesem Event ist im Gegensatz zum zuvor beschriebenen, dass es sich um eine Erweiterung des Events `CairngormEvent` handelt, also nicht vom `FlashEvent` abgeleitet ist. Ein `CairngormEvent`

besitzt die `dispatch()`-Methode, also die Möglichkeit, sich selbst feuern zu können.
Der grundlegende Unterschied zwischen Flash- und Cairngorm-Events besteht darin, dass Flash-Events innerhalb der Display-Liste „hoch-bubblen“ (durchgereicht werden) und Cairngorm-Events dagegen direkt gegen den `FrontController` gefeuert werden.

Nachdem im `ZwickrEvent` nun ein Parameter gesetzt wurde, feuert es sich selbst. Man kann diese Aufeinanderfolge verschiedenartiger Events mit *heterogener Event-Kette* bezeichnen, da ein `FlashEvent` ein weiteres `CairngormEvent` auslöst.

Im weiteren Verlauf wird das gefeuerte `ZwickrEvent` vom `FrontController` abgefangen und über die `addCommand()-Methode` auf das dafür vorgesehene Command `ChangeDatenquaderCommand` gemappt.

```
private function setupCommands():void {
  addCommand( ZwickrEvent.CHANGE_DATENQUADER,
    ChangeDatenquaderCommand );
}
```

Quellcode 15 - Command Mapping durch den FrontController

Das Command kommuniziert nun über den `ModelLocator` mit dem `GeneralModel` und veranlasst die Änderung des spezifischen Models über die `execute()`-Methode.

```
public function execute( event:CairngormEvent ):void {
  model.generalModel.currentMainState = event.data;
}
```

Quellcode 16 - Command wird ausgeführt

Da zwischen der View `MainContent` und dem `GeneralModel` eine Datenbindung besteht, passt sich die View unmittelbar an die Änderung des Models an.

```
currentState="{ model.generalModel.currentMainState }"
```

Quellcode 17 - Datenbindung zwischen MainContent und GeneralModel

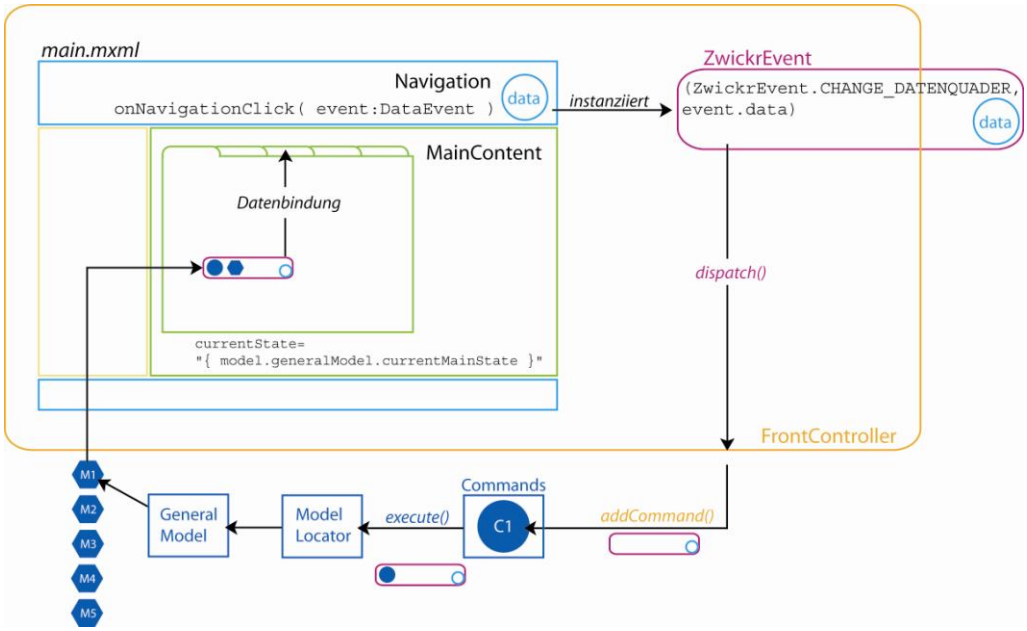


Abb. 32 - Kommunikation zwischen Navigation und MainContent

6.1.6 Mockup

Das Datenmodell von GENESIS-Online ist bereits aus Kapitel 4.1 bekannt. In Kapitel 5.3 wurde des Weiteren herausgearbeitet, welche Möglichkeit einer Backend-Schnittstelle für die Nutzung externer Daten besteht. An dieser Stelle folgt nun die Beschreibung, wie diese Daten innerhalb der Anwendung Zwickr verwendet werden:

Zu Zeiten des Prototyping wird auf eine Datenbankanbindung verzichtet und in der Anwendung Zwickr mit sogenannten *Mockup*-Daten gearbeitet, also lokal in der Applikation gespeicherten Daten.

Aus dem vielseitigen Angebot des Statistischen Bundesamtes Deutschland werden fünf konkrete Tabellen verwendet, die im Vorfeld sorgfältig ausgewählt und analysiert wurden. Da sich nicht jede Datengrundlage für die Darstellung in einem beliebigen Diagramm eignet, sind bestimmte Parameter festgelegt worden. Diese Parameter haben sich während der Analyse dieser Daten herauskristallisiert, mithilfe derer ein oder mehrere Diagrammtypen für die Visualisierung als geeignet bewertet werden können. Zu diesen Parametern zählen beispielsweise Zeit (Linien- oder Balkendiagramm), Region (thematische Karte) und Anteile eines Ganzen (Kreisdiagramm).

Wie dem Kapitel 2 des Lastenheftes [vgl. Anlage A2] zu entnehmen ist, handelt es sich bei den ausgewählten Datensätzen um folgende:

- Spätaussiedler (Stat.Bund.Nr.: 12711 – 0010)
- Bundestagswahl (Stat.Bund.Nr.: 14111 – 0001)
- Kindergeld (Stat.Bund.Nr.: 22911 – 0002)
- Ankünfte und Übernachtungen in Beherbergungsbetrieben (Stat.Bund.Nr.: 45412 – 0011)
- Ausgaben für den Umweltschutz (Stat.Bund.Nr.: 85411 – 0001)

Die Mockup-Daten dieser fünf Datensätze sind in ihrer jeweiligen Klasse in einer `ArrayCollection` angelegt.

```
package de.zwickr.mockup
{
  import mx.collections.ArrayCollection;

  public class KindergeldDaten
  {
    public static var data:ArrayCollection = new
      ArrayCollection(
        [

. . .

         {jahr:       2006
          , kinder:   15234
          , beitraege: 29787
         }
        ,{
          jahr:       2007
          , kinder:   15024
          , beitraege: 29262
         }

        ]

      );
    public function KindergeldDaten() {
    }
  }
}
```

Quellcode 18 - Ausschnitt der Klasse KindergeldDaten.as

In Quellcode 18 ist der Aufbau einer Klasse verkürzt auf zwei Objekte der `ArrayCollection` am Beispiel der Kindergeld-Daten dargestellt. Dieser Aufbau der Mockup-Daten ermöglicht es, die statischen Daten zu

einem späteren Zeitpunkt problemlos auszutauschen, um sie beispielsweise durch ein dynamisches Backend zu ersetzen.

6.2 Interaktion

Nachdem sich die letzten Kapitel mit den Hauptbestandteilen der Applikation und der Kommunikation dieser untereinander beschäftigt haben, werden im Folgenden die Interaktionen innerhalb des `MainContent`, also der Views und ihrer zugehörigen Menüs, behandelt [vgl. Abb. 29]. Dies geschieht am Beispiel zweier Tabellen innerhalb der Zwickr-Applikation.

6.2.1 Interaktion zwischen Menü und View

Die Beschreibung der Interaktionen zwischen Menü und View erfolgt am Beispiel der Spätaussiedler-Daten:

Als Ausgangssituation besteht die View `MainContent` standardmäßig aus dem Liniendiagramm (engl. Linechart) und dem noch geschlossenen Spätaussiedler-Menü. Bezeichnet sind diese mit `SpaetaussiedlerLineChart` und `SpaetaussiedlerMenu`. Dargestellt werden die Spätaussiedler Deutschlands unterteilt in sechs verschiedene Altersgruppen im Zeitraum zwischen 1997 und 2006 [vgl. Abb. 33]. An dieser Stelle ist das Diagramm bereits interaktiv und zeigt bei Mouseover mit dem Cursor Detailwerte an.

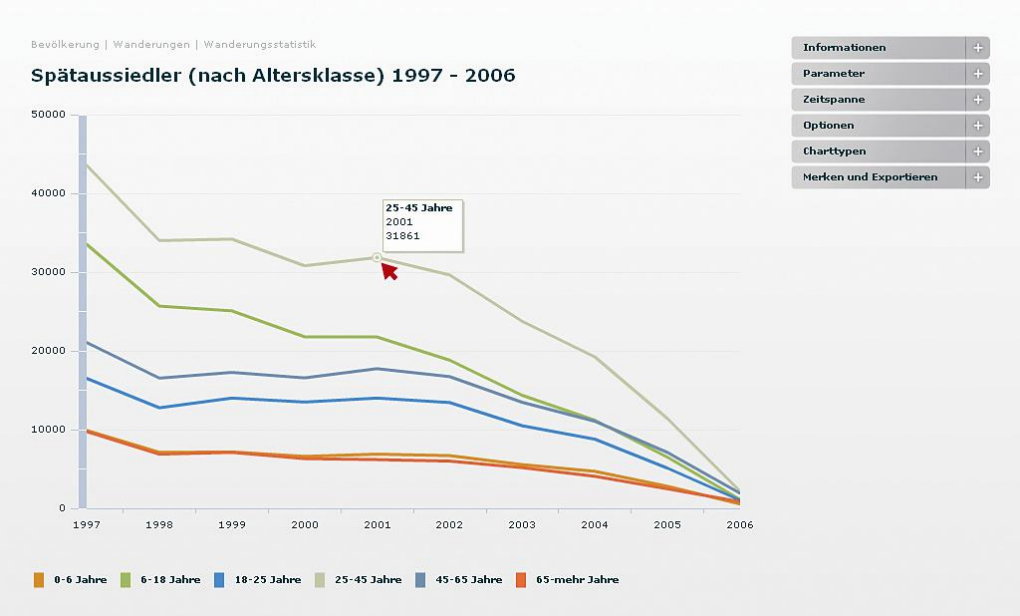


Abb. 33 - SpaetaussiedlerLineChart und SpaetaussiedlerMenu

Diese Funktion `showDataTips` ist in der `LineChart`-Komponente von Flex verankert und erwartet zum Aktivieren oder Deaktivieren einen Boolschen Wert (`true` oder `false`) [vgl. Quellcode 19].

```
<mx:LineChart id="lineChart"
    left="0" top="70" right="0" bottom="60"
    showDataTips="true"
    seriesFilters="{ model.generalModel.seriesFilter }"
    dataProvider="{ model.spaetaussiedlerModel.currentData }"
    series="{ model.spaetaussiedlerModel.currentLineSeries }">
  <mx:horizontalAxis>
    <mx:CategoryAxis categoryField="jahr"/>
  </mx:horizontalAxis>
</mx:LineChart>
```

Quellcode 19 - Instanziierung der LineChart

6.2.1.1 Parameter

Der erste für die Datenvisualisierung relevante Menüpunkt bietet eine Parameterauswahl über mehrere Checkboxen. Die Anzahl der Graphen im Chart richten sich nach der Auswahl im Menü [vgl. Abb. 34].

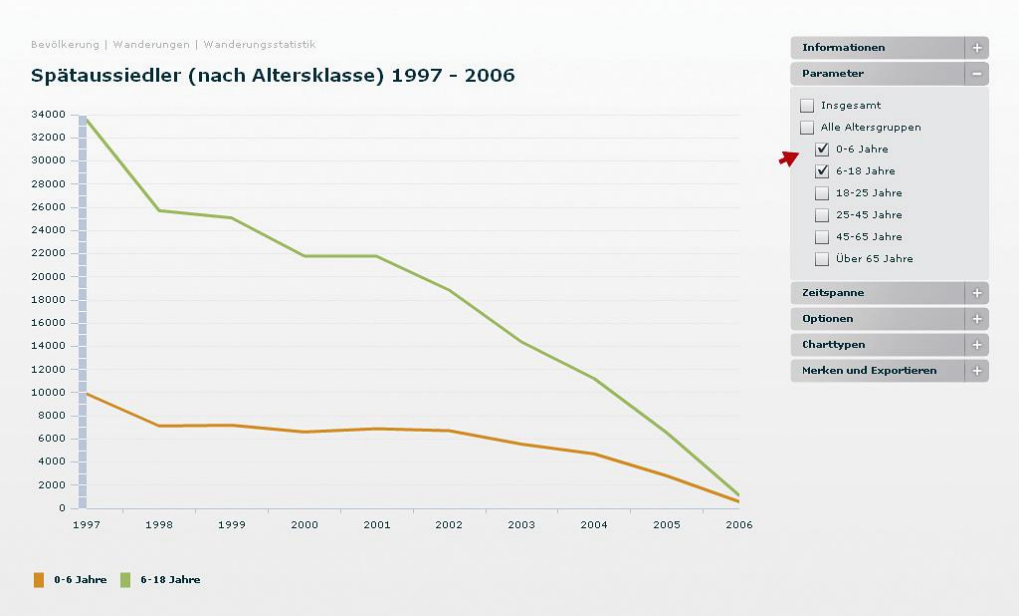


Abb. 34 - LineChart mit Parameterauswahl

Das Menü setzt sich aus sechs Instanzen des `CollapsableMenuItem` zusammen. Die Checkboxen des oben beschriebenen Menüpunktes weisen über ihre Eigenschaft `selected` eine Datenbindung zum `SpaetaussiedlerModel` auf.

```
<components:CollapsableMenuItem label="Parameter">
    <components:content>
      <mx:Panel width="100%" styleName="accordionPanel">
        <mx:CheckBox id="ageInsgesamtCB" label="Insgesamt"
            selected="{model.spaetaussiedlerModel.
                ageInsgesamtCBSelected }"
            change="controller.onAgeCheckBoxChange(event)" />

. . .

      </mx:Panel>
    </components:content>
  </components:CollapsableMenuItem>
```

Quellcode 20 - Eigenschaften einer Checkbox

Über das `change`-Event wird der Event Handler `onAgeCheckboxChange` des Controllers aufgerufen. Dort werden über eine `if`-Bedingung alle ausgewählten Checkboxen herausgefiltert, und über die Methode `push()` werden die entsprechenden Parameter an das Array `selectedAges` übergeben. Dieses wird dem Model zugewiesen.

```
public function onAgeCheckBoxChange( event:Event ):void {
  var selectedAges:Array = new Array();

    if ( view.ageInsgesamtCB.selected ) {
      selectedAges.push( "insgesamt" );
      model.spaetaussiedlerModel.ageInsgesamtCBSelected =
          true;
    } else {
      model.spaetaussiedlerModel.ageInsgesamtCBSelected =
          false;
    }


. . .

    model.spaetaussiedlerModel.selectedAges = selectedAges;
}
```

Quellcode 21 - Event Handler onAgeCheckBoxChange()

Das Model liest nun das Array aus und aktualisiert seinen Datenbestand, um wiederum über die `push()`-Methode Parameter an das Array `currentLineSeries` zu übergeben.

```
private function createLineSeries():void {
  var series:LineSeries;
  var fx:SeriesEffect;
  currentLineSeries = new Array();

  for each ( var age:String in _selectedAges ) {
    series = new LineSeries();
    series.setStyle( "form", "segment");
    fx = new SeriesInterpolate();
    series.setStyle( "showDataEffect", fx );
    switch ( age ){

. . .
    }
    currentLineSeries.push( series );
  }
}
```

Quellcode 22 - Methode createLineSeries()

Die folgende Abbildung veranschaulicht den Interaktionskreislauf zwischen Checkboxen und View:

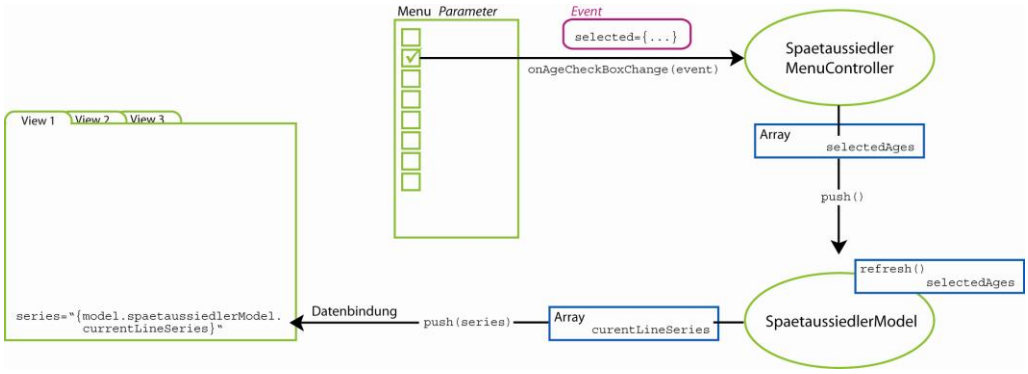


Abb. 35 - Menü Parameter

6.2.1.2 Zeitspanne

Standardmäßig zeigt das Diagramm der Tabelle Spätaussiedler einen Zeitraum von neun Jahren an (1997-2006). Für die zeitliche Eingrenzung der Darstellung enthält das Menü einen sogenannten *DualSlider*, mit dessen Hilfe der Benutzer den Zeitraum frei wählen kann. Dabei steht ihm frei, den linken, rechten oder auch beide Slider zu verschieben. Die kleinste Spanne, die erreicht werden kann, ist ein Jahr.

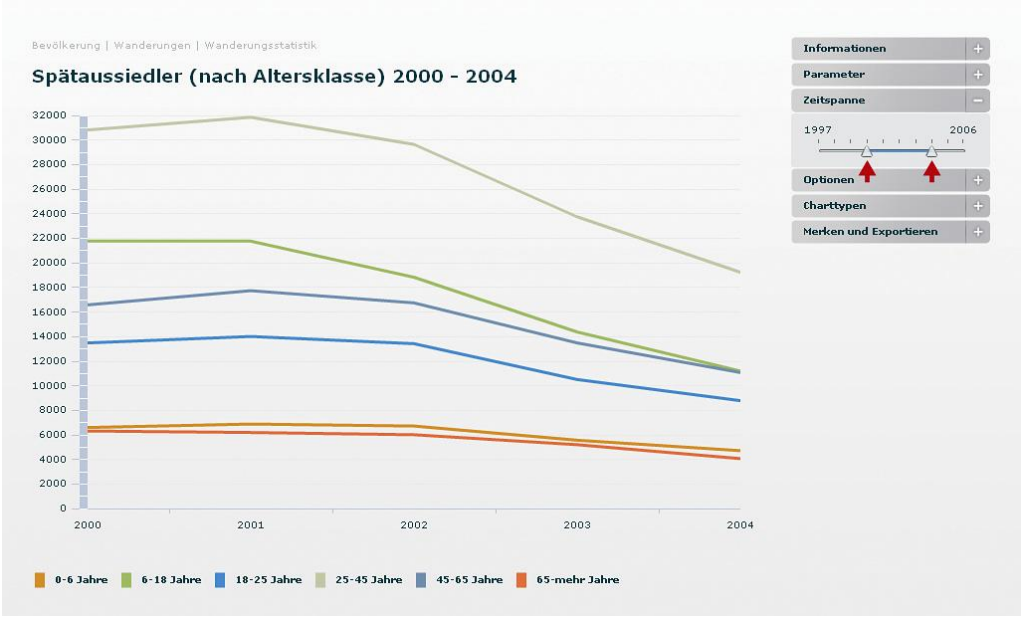


Abb. 36 - LineChart mit DualSlider

Über das `change`-Event wird der Event Handler `onSliderChange()` des Controllers aufgerufen.

```
<flexlib:HSlider width="100%" minimum="1997" maximum="2006"
    values="[1997, 2006]"
    snapInterval="1"
    liveDragging="true"
    showTrackHighlight="true"
    labels="[1997, 2006]"
    tickInterval="1"
    thumbCount="2"
    dataTipFormatFunction="{controller.sliderDataTipFunction}"
    change="controller.onSliderChange(event)" />
```

Quellcode 23 - Instanziierung des HSlider als Zeitslider

Dort werden die Werte des aktuell ausgewählten Start- und Endjahres für die Variablen des Models SpaetaussiedlerModel `startYear` und `endYear` gesetzt.

```
public function onSliderChange( event:SliderEvent ):void {
  model.spaetaussiedlerModel.startYear = HSlider(
      event.currentTarget ).values[0];
  model.spaetaussiedlerModel.endYear = HSlider(
      event.currentTarget ).values[1];
}
```

Quellcode 24 - Event Handler onSliderChange()

Per Getter- und Setter-Funktion im `SpaetaussiedlerModel` werden die gesetzten Werte ein- und ausgelesen. Die Datenbindung zur View über den `dataProvider` sorgt dann für die Aktualisierung der Diagrammansicht [vgl. Quellcode 19].

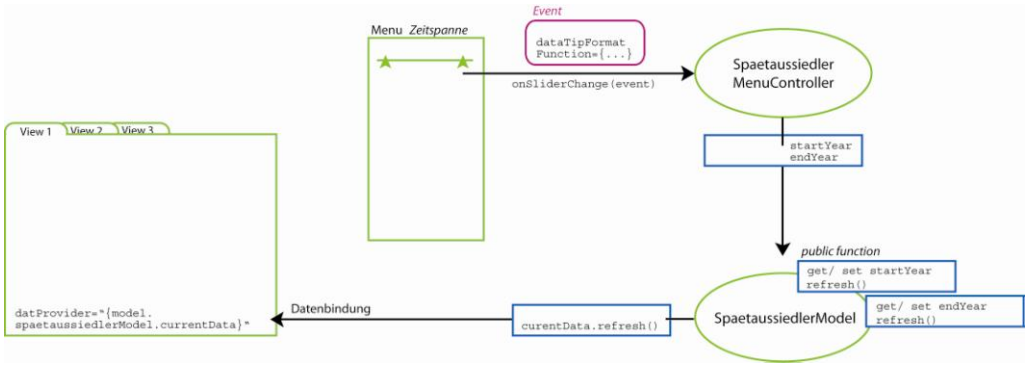


Abb. 37 - Menü Zeitspanne

6.2.1.3 Charttypen

Neben dem bisher erläuterten Liniendiagramm existieren für den Datenbestand der Spätaussiedler zwei weitere Diagrammtypen: das Balkendiagramm (engl. Barchart) und das Kreisdiagramm (engl. Piechart). Zum Wechsel zwischen den drei Diagrammtypen enthält das Menü den Unterpunkt *Charttypen*. Dahinter verbergen sich drei Schaltflächen, die jeweils über das `click`-Event die Variable `currentMainState` vom Typ `String` im `SpaetaussiedlerModel` neu setzen. Hierzu werden Konstanten verwendet, um die Codehilfe zu aktivieren und eine Konsistenz zu gewährleisten.

```
<mx:Button icon="@Embed(source='assets/pics/line_icon.png')"
  click="model.spaetaussiedlerModel.currentMainState=
    SpaetaussiedlerMain.LINE_CHART" />
<mx:Button icon="@Embed(source='assets/pics/bar_icon.png')"
  click="model.spaetaussiedlerModel.currentMainState=
    SpaetaussiedlerMain.BAR_CHART" />
<mx:Button icon="@Embed(source='assets/pics/pie_icon.png')"
  click="model.spaetaussiedlerModel.currentMainState=
    SpaetaussiedlerMain.PIE_CHART" />
```

Quellcode 25 - Drei Instanzen des Button zum Charttypwechsel

```
public var currentMainState:String =
    SpaetaussiedlerMain.LINE_CHART;
```

Quellcode 26 - Variable currentMainState vom Typ String

Über die Datenbindung zur View `SpaetaussiedlerMain` wird das aktuell ausgewählte Diagramm angezeigt. Innerhalb des Balkendiagramms sind dem Benutzer die gleichen Interaktionsmöglichkeiten gegeben wie im zuvor beschriebenen Liniendiagramm.

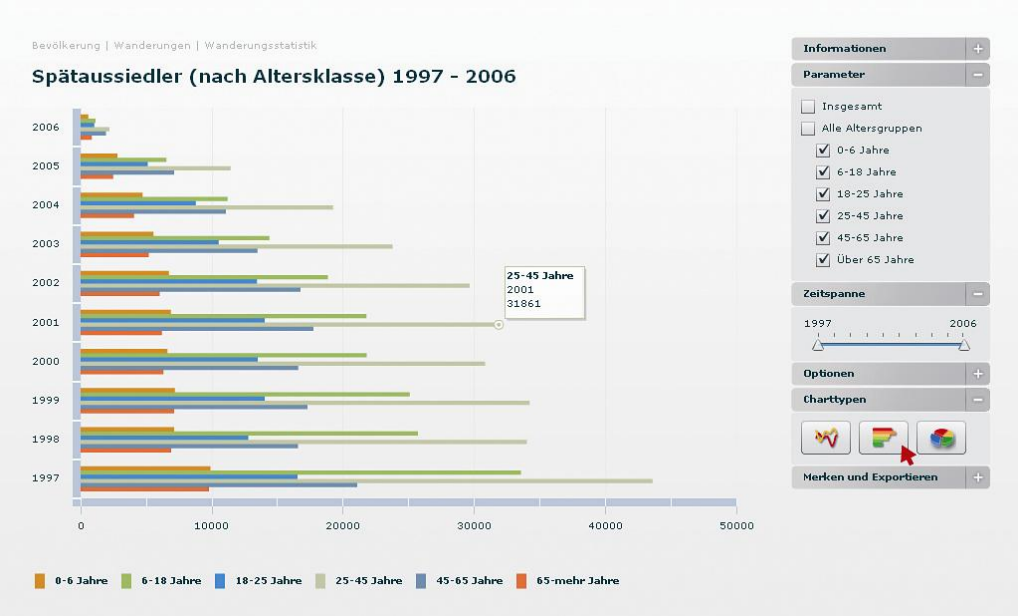


Abb. 38 - Charttyp BarChart

Das Kreisdiagramm hingegen weist eine Besonderheit auf. Da das Kreisdiagramm seine Werte nur für jeweils ein Jahr darstellen kann, macht an dieser Stelle der in den anderen Diagrammformen verwendete Zeitslider keinen Sinn. Dem Benutzer wird stattdessen eine Zeitauswahl in Form des *NumericStepper* angeboten. Er kann die Zeitauswahl somit schrittweise vorwärts oder rückwärts verändern.

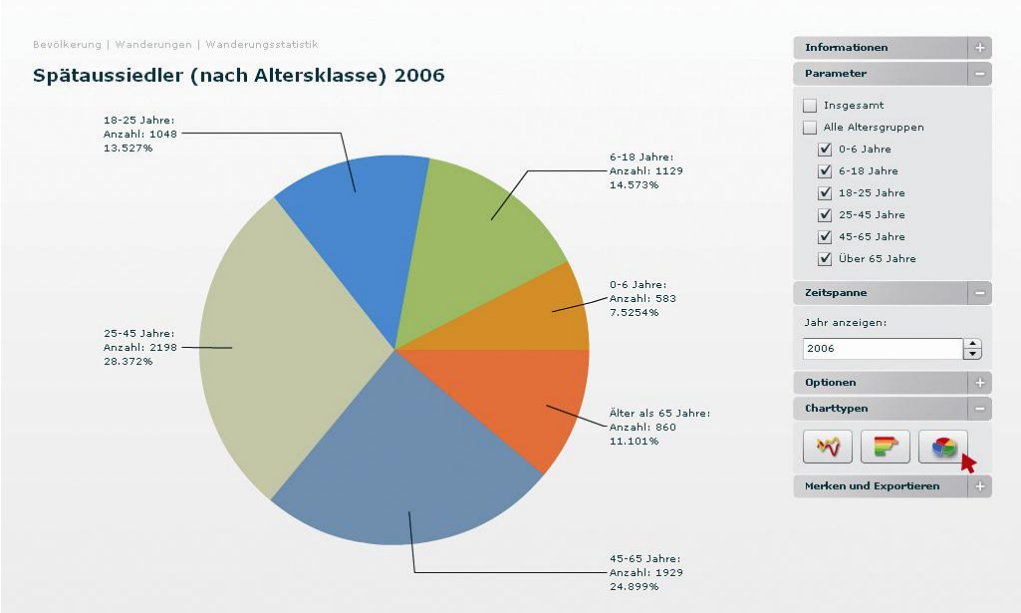


Abb. 39 - Charttyp PieChart

Das `change`-Event des `NumericStepper` sorgt dafür, dass die Setter-Funktionen im `SpaetaussiedlerModel` das aktuell ausgewählte Jahr `pieChartYear` einlesen und die View per Datenbindung aktualisiert wird.

```
<mx:NumericStepper width="100%" minimum="1997" maximum="2006"
    value="2006"
  change="model.spaetaussiedlerModel.pieChartYear=
      event.currentTarget.value" />
```

Quellcode 27 - NumericStepper

6.2.2 Interaktion innerhalb der Multibox

In Kapitel 6.2.1 wurde die Interaktivität zwischen View und Menü erläutert. Die folgenden Abschnitte widmen sich der Interaktion einzelner Diagramme untereinander innerhalb einer View, der sogenannten *Multibox*. Dies erfolgt am Beispiel der Bundestagswahlen-Daten:

Als Ausgangssituation besteht die View `MainContent` standardmäßig aus der Multibox und dem geschlossenen Bundestagswahlen-Menü, welches für die Beschreibung der Interaktivität der Multibox vernachlässigt wird. Bezeichnet ist die Multibox mit `Bundestagswahlen-Drilldown`. Dargestellt werden die Ergebnisse der vergangenen Bundestagswahlen Deutschlands in drei gleichzeitig sichtbaren Diagrammformen: Abb. 40 zeigt verdeutlichend den Aufbau der Multibox mithilfe der Einteilung in Quadranten. So befindet sich im zweiten Quadranten ein Liniendiagramm, welches die Ergebnisse zwischen 1994 und 2005 visualisiert. Der dritte Quadrant beinhaltet ein Kreisdiagramm mit der standardmäßigen Ansicht der Wahlergebnisse des Jahres 2005. Quadrant eins und vier ergeben zusammen das Feld der thematischen Karte von Deutschland, die in ihre 16 Bundesländer eingeteilt ist. In der Standardansicht der Karte wird die Verteilung der beiden Wahlsieger des Jahres 2005 SPD und CDU durch rote oder blaue Färbung der jeweiligen Bundesländer visualisiert.

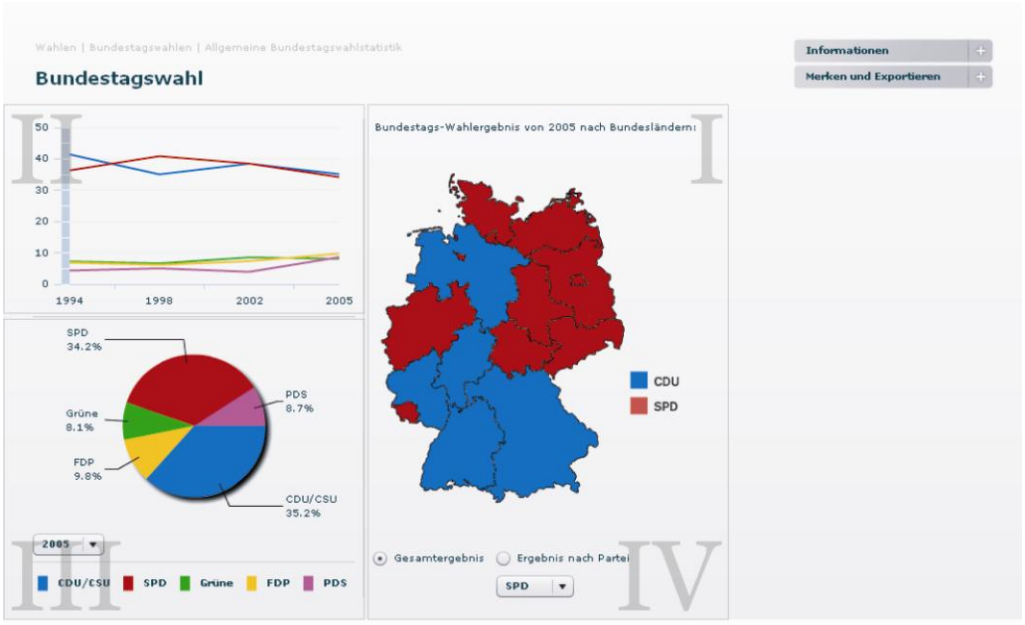


Abb. 40 - Multibox Bundestagswahlen

Zwischen diesen drei Bereichen herrschen gewisse Interaktionsmöglichkeiten, die dem Benutzer zu einer noch schnelleren Auffassung der visualisierten Zusammenhänge verhelfen sollen. Die Beschreibung der einzelnen Funktionalitäten erfolgt in den kommenden Abschnitten.

6.2.2.1 Funktionen des Liniendiagramms

Das Liniendiagramm besitzt ein `itemClick`-Event, über das nach einer Benutzerinteraktion, speziell dem Klicken eines Wertepunktes, der Event Handler `onLineChartItemClick()` des Controllers aufgerufen wird.

```
<mx:LineChart id="lineChart"
    width="100%" height="70%"
    showDataTips="true"
    seriesFilters="{ model.generalModel.seriesFilter }"
    dataProvider="{ model.bundestagswahlenModel.currentData }"
    series="{ model.bundestagswahlenModel.currentLineSeries }"
    itemClick="controller.onLineChartItemClick(event)" />
```

Quellcode 28 - Instanziierung der Charting Component LineChart

Der Controller prüft über eine `switch-case`-Bedingung, welches Jahr dabei ausgewählt wurde. Anschließend wird das ausgewählte Jahr `selectedYear` auf die ComboBox `myPieChartYearComboBox` des Kreisdiagramms gemappt.

```
public function onLineChartItemClick( event:ChartItemEvent ):
    void {
  var selectedYear:uint = uint( event.hitData.item.jahr );
  model.bundestagswahlenModel.pieChartYear = selectedYear;
  deselectPieChartItem();

  var index:uint;
    switch( selectedYear ) {
      case 1994:      index = 0;      break;
      case 1998:      index = 1;      break;
      case 2002:      index = 2;      break;
      case 2005:      index = 3;      break;
    }
  view.myPieChartYearComboBox.selectedIndex = index;
}
```

Quellcode 29 - Event Handler onLineChartItemClick()

Über das `change`-Event der `ComboBox` wird der Event Handler `onPieChartYearComboBoxChange()` des Controllers aufgerufen. Dieser Event Handler bewirkt, dass bei Änderung der `ComboBox` der Wert des aktuell ausgewählten Jahres für die Getter- und Setter-Funktionen im `BundestagswahlenModel` gesetzt werden.

```
public function onPieChartYearComboBoxChange( event:ListEvent)
    :void {
  var selectedYear:uint = uint( ComboBox( event.
      currentTarget ).selectedLabel );
  model.bundestagswahlenModel.pieChartYear = selectedYear;
  deselectPieChartItem();
  view.gesamtButton.selected=true;
}
```

Quellcode 30 - Event Handler onPieChartYearComboBoxChange()

Daraufhin erfolgt die Aktualisierung des Kreisdiagramms über die Methode `createPieSeries()`.

```
public function set pieChartYear( year:int ):void {
   _pieChartYear = year;
   createPieSeries();
}

public function get pieChartYear():int {
   return _pieChartYear;
}
```

Quellcode 31 - Getter- und Setter-Funktion

Aber nicht nur das Kreisdiagramm reagiert auf die Benutzerinteraktion mit dem Liniendiagramm, sondern auch die Karte, wie man im Vergleich von Abb. 40 zu Abb. 41 erkennen kann.

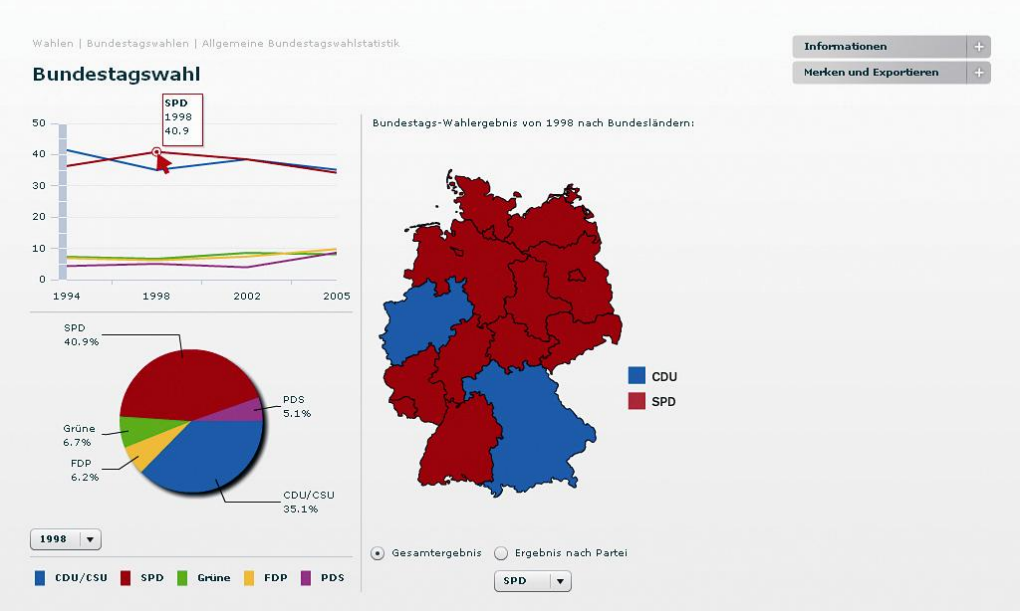


Abb. 41 - Funktionen des Liniendiagramms

Die Aktualisierung dieser View erfolgt über die Methode `deselectPieChartItem()` des Controllers [vgl. Quellcode 30], die dafür sorgt, dass die entsprechende Karte zum ausgewählten Jahr angezeigt wird. Dabei wird das ausgewählte Jahr `pieChartYear` in eine Variable `year` vom Typ `uint` geschrieben. Über ein *Dictionary* ist jedem `year` ein String zugeordnet, der dem Dateinamen der Karte entspricht.

```
public function deselectPieChartItem():void {
  var arr:Array = [];
  view.pieChart.series[0].perWedgeExplodeRadius = arr;

  view.mySwfMap.visible = false;
  var dict:Dictionary = new Dictionary();
  dict["1994"] = "map1994.swf";
  dict["1998"] = "map1998.swf";
  dict["2002"] = "map2002.swf";
  dict["2005"] = "map2005.swf";

  var year:uint = model.bundestagswahlenModel.pieChartYear;
  var swfName:String = dict[ year ];
  view.mySwfMap.source = "assets/swf/"+swfName;
  view.mySwfMap.visible = true;
}
```

Quellcode 32 - Methode deselectPieChartItem()

6.2.2.2 Funktionen des Kreisdiagramms

Während es dem Benutzer über das Liniendiagramm möglich ist, die Ansicht der einzelnen Diagramme in zeitlicher Hinsicht zu verändern, bewirkt ein Klick über das Kreisdiagramm eine Veränderung hinsichtlich der Partei. Dabei entsteht ein Abstand der ausgewählten Ecke des Kreisdiagramms zum restlichen Diagramm [vgl. Abb. 42].

Ähnlich wie das Liniendiagramm besitzt auch das Kreisdiagramm ein `itemClick`-Event, über die der Event Handler `onPieChart-ItemClick()` des Controllers aufgerufen wird.

```
<mx:PieChart id="pieChart" width="100%" height="100%"
  seriesFilters="{ model.generalModel.seriesFilter }"
  dataProvider="{ model.bundestagswahlenModel.currentPieData}"
  selectionMode="single"
  itemClick="controller.onPieChartItemClick(event)" />
```

Quellcode 33 - Instanziierung der Charting Component PieChart

```
public function onPieChartItemClick( event:ChartItemEvent):
    void {
  var arr:Array = [];
  arr[event.hitData.chartItem.index] = 0.2;
  PieChart( event.currentTarget).series[0].
      perWedgeExplodeRadius = arr;

  var party:String = event.hitData.item.party;
  var partyId:String = event.hitData.item.partyId;
  var year:uint = model.bundestagswahlenModel.pieChartYear;
  showMapByPartyAndYear( partyId, year );
}
```

Quellcode 34 - Event Handler onPieChartItemClick()

Die Aktualisierung der Karte erfolgt über die Methode `showMapBy-PartyAndYear()` des Controllers.

```
public function showMapByPartyAndYear( partyId:String,
    year:uint ):void {
  view.mySwfMap.visible = false;

  var dict:Dictionary = new Dictionary();
  dict["spd1994"] = "mapSpd94.swf";
  dict["spd1998"] = "mapSpd98.swf";
  dict["spd2002"] = "mapSpd02.swf";
  dict["spd2005"] = "mapSpd05.swf";


. . .

  var swfName:String = dict[ partyId+year ];

  view.mySwfMap.source = "assets/swf/"+swfName;
  view.mySwfMap.visible = true;
}
```

Quellcode 35 - Methode showMapByPartyAndYear()

Zwei Parameter werden von dieser Methode erwartet: `partyID` vom Typ `String` und `year` vom Typ `uint`. Der Parameter `partyID` erhält seinen Wert aus dem `BundestagswahlenModel`.

```
currentPieData.addItem(
  {
    party:     "FDP"
    , partyId:  "fdp"
    , value:  tempDataset.fdp_prozent
  }
);
```

Quellcode 36 - Ausschnitt der Funktion createPieSeries()

Der Parameter `year` wird ebenfalls über das Model mit einem Wert gefüllt.
Und so erhält die rechte Seite der Multibox eine Karte, die die Wahlergebnisse einer bestimmten Partei im ausgewählten Jahr visualisiert:

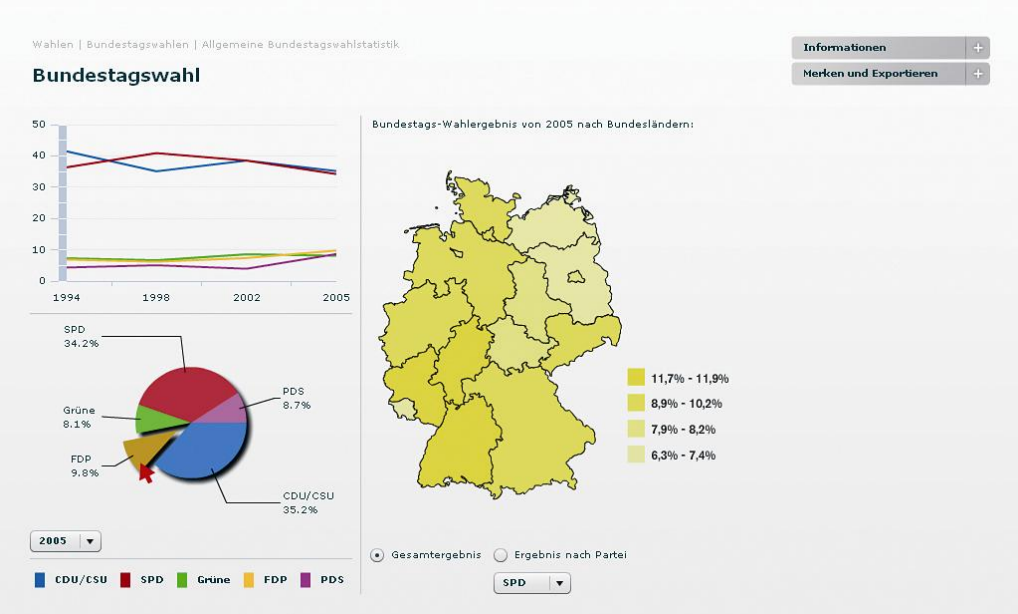


Abb. 42 - Funktionen des Kreisdiagramms

6.2.2.3 Funktionen der Karte

Wie schon aus Kapitel 6.2.2.2 bekannt, können die Bundestagswahlergebnisse der einzelnen Parteien in einer Karte visualisiert werden. Um zu dieser Ansicht zu gelangen, gibt es neben der Möglichkeit über das Kreisdiagramm auch eine Funktion innerhalb der Karte selbst. Zwei `RadioButton`-Komponenten regeln die Ansicht in Verbindung mit den anderen Diagrammen, bzw. mit der ComboBox `myMapPartyComboBox` innerhalb der Karte. Wird der `RadioButton` mit der Beschriftung *Ergebnis* nach Partei aktiviert und über die darunter positionierte `ComboBox` eine Partei ausgewählt, so passt sich die View an. Dabei kommt das `change`-Event der `ComboBox` zum Tragen.

```
<mx:ComboBox right="400" id="myPieChartYearComboBox"
  change="controller.onPieChartYearComboBoxChange(event)"
  selectedIndex="2005">
    <mx:dataProvider>
      <mx:ArrayCollection>
        <mx:Object label="1994" />
        <mx:Object label="1998" />
        <mx:Object label="2002" />
        <mx:Object label="2005" />
      </mx:ArrayCollection>
    </mx:dataProvider>
</mx:ComboBox>
```

Quellcode 37 - ComboBox myPieChartYearComboBox

Sie ruft den Event Handler `onMapComboBoxChange()` des Controllers auf, in dem die Methode `showMapByPartyAndYear()` [vgl. Quellcode 35] aufgerufen wird.

```
public function onMapComboBoxChange( event:ListEvent ):void {
    view.parteiButton.selected=true;
  var selectedFilters:String = String( ComboBox(event.
      currentTarget ).selectedLabel );

  showMapByPartyAndYear(view.myMapPartyComboBox.
      selectedItem.data, int(view.myPieChartYearComboBox.
      selectedLabel));
}
```

Quellcode 38 - Event Handler onMapComboBoxChange()

Der erste Parameter `partyID` vom Typ `String`, wird mit einem Wert des dataProvider der `ComboBox myMapPartyComboBox` gefüllt. Der Wert des zweiten Parameters `year` vom Typ `uint` stammt aus dem dataProvider der `ComboBox myPieChartYearComboBox` [vgl. Quellcode 37].

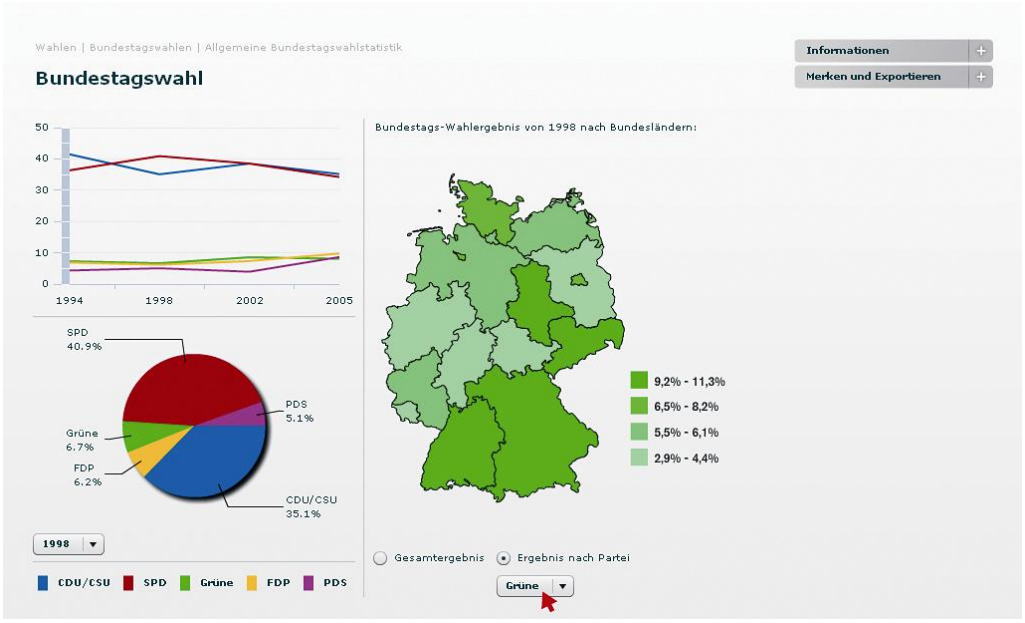


Abb. 43 - Funktionen der Karte I

Der zweite RadioButton in der Karte mit der Beschriftung Gesamtergebnis führt bei Aktivierung dazu, dass zu dem aktuell ausgewählten Jahr die Verteilung der beiden Wahlsieger SPD und CDU durch rote oder blaue Färbung der jeweiligen Bundesländer visualisiert werden:

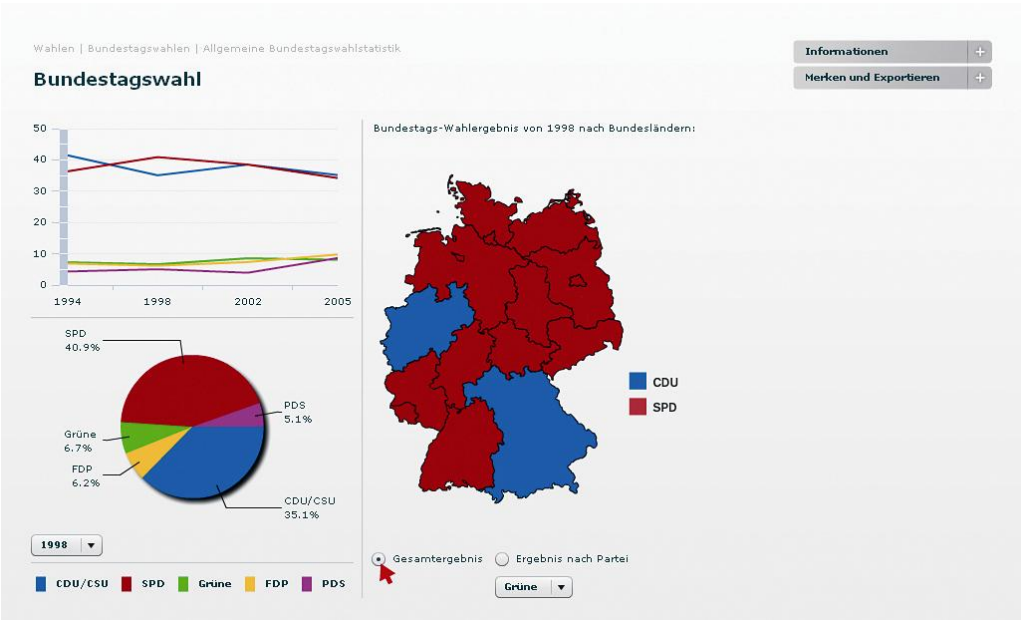


Abb. 44 - Funktion der Karte II

Dies wird durch das `click`-Event des RadioButton `gesamtButton` erreicht, die die Methode `deselectPieChartItem()` aufruft [vgl. Quellcode 32].

7 Bewertung und Ausblick

Die vorliegende Arbeit setzte sich im Zusammenhang mit der Entwicklung einer Rich Internet Application zur interaktiven Visualisierung statistischer Massendaten mit zwei Aufgabenbereichen auseinander. Zum einen sollte eine technische Konzeption mit Diskussion um mögliche Technologien erarbeitet werden. Zum anderen sollten diese konzeptionell erzielten Ergebnisse im praktischen Teil der Softwareentwicklung angewandt werden.
Rückblickend werden nun die Ergebnisse im Folgenden einem kurzen Fazit unterzogen und Perspektiven für mögliche Weiterentwicklungen aufgezeigt.

Die Aufgabenstellung, eine Recherche im Hinblick auf technologische und wirtschaftliche Aspekte durchzuführen und anschließend eine Technologieentscheidung für die Entwicklung einer Rich Internet Application zu fällen, wurde ganzheitlich erfüllt. Die Softwareentwicklung zur interaktiven Visualisierung statistischer Massendaten fand im Rahmen des Prototyping statt und ist in dem in der vorliegenden Arbeit beschriebenen Status voll funktionsfähig. Die Realisierung dieser Anwendung erforderte die selbständige und systematische Einarbeitung in ein gänzlich neues Themengebiet, wobei für auftretende Probleme zeitnahe Lösungen gefunden werden mussten. Durch die Projektarbeit innerhalb eines Betriebes, dem Atelier für Mediengestaltung, herrschten authentische Arbeitsbedingungen, so dass Erfahrungen im Bezug auf Praxisnähe gesammelt werden konnten.

Während zunächst die Überlegung bestand, die geplante Anwendung evtl. mit betriebsinternen Mitteln, beispielsweise mit PHP, zu realisieren, erbrachte die Recherche nach einer geeigneten Technologie schnell das Ergebnis, dass lediglich die in dieser Arbeit vorgestellten Technologien Adobe Flex und Microsoft Silverlight den Ansprüchen an eine RIA gewachsen sein würden. So erwies sich die Entwicklung auf Basis von Flex 3 als sehr zufriedenstellend und bot schon nach kurzer Einarbeitungszeit vielversprechende Ergebnisse. Etwas schwieriger dagegen

gestaltete sich das Kennenlernen und Anwenden von Design-Pattern und dem Mikroarchitektur-Framework Cairngorm. Hierbei handelte es sich keinesfalls um Basiswissen und bedurfte daher ein wenig Anlaufzeit. Im Nachhinein kann jedoch behauptet werden, dass der Austausch innerhalb des Projektteams zum Verständnis der abstrakten Zusammenhänge führte und es dadurch möglich war, eine agile Anwendung und damit den Grundstein für die Weiterentwicklung zu schaffen.

Im Hinblick auf mögliche Weiterentwicklungen der Applikation wurden bereits Ideen und Ansätze erarbeitet, die es weiter zu entwickeln gilt. Unter anderem könnte eine Datenbank entstehen, die eine Schnittstelle zwischen der Anwendung und beliebigen anderen Datenbanken darstellt. Während die Visualisierung bislang mit den Flex Charting-Komponenten durchgeführt wurde, bestehen Möglichkeiten der Anbindung fremder Charting-Komponenten. Dazu gehören unter anderem die Fusion Charts[7] und Komponenten von ILOG Elixir[8], Kap Lab[9] oder Papervision 3D[10]. Des Weiteren besteht durch die Adobe Integrated Runtime (kurz AIR), die mit Flex und Flash zur sogenannten *Adobe Flash Platform* gehört, die Möglichkeit aus der Applikation eine Desktop-Anwendung zu erstellen.

Die erfolgreiche Konzeption und Umsetzung der Rich Internet Application erforderte neben der fachlichen Qualifikation die Fähigkeit, bisher unbekannte Zusammenhänge erfassen und verinnerlichen zu können und auf den Ergebnissen aufbauend neue Strategien sowie Lösungsansätze erarbeiten zu können. Das produktive Endergebnis dieser Arbeit ist zur vollsten Zufriedenheit des Unternehmens ausgefallen und wird in naher Zukunft als repräsentativer Prototyp zu Präsentationszwecken und als technische Grundlage für die Weiterentwicklung dienen.

---

[7] http://www.fusioncharts.com

[8] http://www.ilog.com/products/ilogelixir

[9] http://lab.kapit.fr/display/kaplabhome/Home

[10] http://www.papervision3d.org

Im Hinblick auf die Zukunft von Rich Internet Applications entwickelt sich ein Trend zum *Cloud Computing*. Clientseitig installierte Programme und Web-Dienste vermischen sich zunehmend oder finden überwiegend online statt. So bietet beispielsweise Adobe unter dem Codenamen *CoCoMo* (aktuell in der Beta-Version) eine Service-Plattform für die Erstellung und Integration von sogenannten *Social Apps* in bestehende Anwendungen, Microsoft macht sein Office online verfügbar und Google APIs ersetzen clientseitig installierte Office-Software.

Es bleibt also spannend im Bereich der Rich Internet Applications.

zwickr - Lastenheft

ENTWICKLUNG EINER „RICH INTERNET APPLICATION“ ZUR INTERAKTIVEN VISUALISIERUNG STATISTISCHER MASSENDATEN

AFM – Atelier für Mediengestaltung

erstellt: 25.03.2009

Zusammenfassung

In diesem Dokument werden die Punkte des Lastenhefts aufgeführt, die zur Konzeption und Realisierung des Prototyps einer Software zur interaktiven Visualisierung statistischer Daten mit automatisiertem Layout gehören.
Abbildungen, die die verschiedenen Funktionalitäten veranschaulichen, befinden sich in den Anlagen und sind über die entsprechenden Verweise zuzuordnen.

Inhaltsverzeichnis

1) Zielbestimmungen – die Entwicklung eines Software-Prototypen

Der Prototyp soll eine schnelle und einfache Recherche und Verarbeitung tagesaktueller Daten, in Form von fünf ausgewählten Datensätzen aus dem Bestand des statistischen Bundesamtes Deutschland, zu interaktiven Infografiken über ein Interface ermöglichen. Die Software ist als lokale Anwendung geplant, sowohl Daten als auch Grafiken liegen clientseitig vor. Die Realisierung erfolgt auf Adobe Flash-Basis, bzw. Adobe Flex, somit wird die Software vom Benutzer plattformunabhängig (Windows, Mac, Linux) genutzt werden können, vorausgesetzt, der Adobe Flash Player (mind. Version 9) ist installiert.

2) Produkteinsatz

Der Software-Prototyp dient ausschließlich Präsentationszwecken. Da er in kurzer Zeit erstellt und hierfür an einigen Stellen mit simulierten Funktionalitäten gearbeitet wird, ist auszuschließen, dass das Folgeprodukt, bzw. die finale Software-Version, darauf aufgebaut werden kann. Der Prototyp soll über User-Tests den Bedarf an einer finalen Version und deren Funktionsumfang ermitteln, daher soll ein vielseitiger Überblick an Funktionalitäten zusammengestellt werden.

Produktbezeichnung	Kurzbeschreibung	Zielgruppen
Prototyp „zwickr“	Einzelne oder miteinander interagierende Infografiken können über fünf verschiedene Navigationswege abgerufen, bearbeitet und weiterverwendet werden.	Keine Einschränkung: Firmen, Schulen, Universitäten, Privatpersonen

3) Produktfunktionen

Hinweis: Die einzelnen Produktfunktionen werden mit einer laufenden Nummer [Bsp.: /LF0100/] versehen, damit alle am Projekt beteiligten Personen jede Funktion eindeutig benennen können. „LF“ steht dabei für „Produktfunktionen des Lastenheftes“.

1. Benutzerfunktionen

1. Die Recherche

Der Benutzer kann das System für seine Recherche über die Navigation nutzen. Dabei werden genau fünf Navigationswege ausgearbeitet sein, die anderen werden nur angedeutet. Dem Benutzer wird suggeriert, er sei eingeloggt, es wird also ein Benutzername angezeigt, und ein Logout-Button (ohne Funktion) deutet seinen Benutzerbereich an. Ebenfalls existiert eine Suchzeile, jedoch ohne Funktion. [Abb. 8, linke Seite]

/LF0100/ Ein Benutzer kann jederzeit über Scrollen über einen Tool-Tipp „Infos anzeigen“ in der Login-Leiste [Abb. 8, linke Seite] einblenden, welche Interaktionsmöglichkeiten er hat. Diese Hilfestellungen werden in halbtransparenten Sprechblasen an der entsprechenden Stelle platziert angezeigt und verschwinden, sobald der Mauszeiger den Tool-Tipp wieder verlässt. Der Tool-Tip wird zu Demonstrationszwecken nur ein paar wenige Punkte zeigen, die vorher festgelegt wurden.

/LF0110/ Ein Benutzer kann jederzeit durch die Hauptnavigation mit neun Kategorien am oberen Full-Screen-Rand navigieren. Per Roll-Over wird ihm angezeigt, auf welchem Button er sich gerade befindet. [Abb. 2]

/LF0120/ Ein Benutzer kann in der Hauptnavigation durch Klick eine Unternavigation (Matrix) ausklappen. Diese fügt sich in das Farbschema der Oberkategorie ein und bildet die einzelnen Unterthemen in verschieden gesättigten Farbtönen der gleichen Farbfamilie ab. [Abb. 3] Da genau fünf Navigationspfade durchlaufen werden können, ist auch hier pro Hauptthema nur ein weiterer Menüpunkt aktiv. Andere Unterthemen werden nur grafisch angedeutet.

/LF0130/ Ein Benutzer kann in der Unternavigation weitere Themeneingrenzungen vornehmen, per Klick erscheinen die zur Thementafel gehörenden Unterthemen ebenfalls in einer Matrix.
[Abb. 4]

/LF0140/ Ein Benutzer kann in der Hauptnavigation eine andere Kategorie auswählen. Damit schließt sich die zuvor geöffnete Unternavigation.

/LF0150/ Ein Benutzer kann über einen Close-Button die Unternavigation schließen.

/LF0160/ Ein Benutzer kann per Drag & Drop in der Unternavigation [Abb. 3] ein für ihn interessantes Haupt-Thema oder eines der Unterthemen als Thementafel (farbiges Feld) [Abb. 4] in seine „Sammlung“ (Leiste am unteren Full-Screen-Rand) aufnehmen. Das Feld in der Unternavigations-Matrix wird in diesem Moment grau, enthält aber weiterhin den Titel des Themas zur späteren Orientierung.
[Abb. 5]

2. Die Sammlung

Am unteren Screen-Rand erstreckt sich über die gesamte Breite eine „Sammlung“, die während der Recherche mit maximal fünf Thementafeln gefüllt werden kann. Die „Sammlung selbst besteht aus zwei nebeneinander liegenden Bereichen - dem, in dem man Thementafeln ablegen kann und dem, in dem man Infografiken zwischenspeichern kann, bevor sie in „myZwickr“ gespeichert werden. Alle Thementafeln, die während der Arbeit mit dem Produkt der „Sammlung“ hinzugefügt werden, sind durch ihre Farben den Hauptkategorien zugeordnet und sortieren sich automatisch thematisch nebeneinander. [Abb. 5, Abb. 6]

/LF0210/ Ein Benutzer kann durch Rollover über die während der Recherche abgelegten Thementafeln nähere Inhalts-Informationen erhalten. Es klappt sich ein kleiner Infotext darüber aus. Eine Tafel beinhaltet die Hauptkategorie und die Unterkategorie.

/LF0220/ Ein Benutzer kann eine Thementafel wieder aus seiner Sammlung entfernen. Dazu klickt er den dafür vorgesehenen Close-Button. Die Thementafel verschwindet aus der „Sammlung“.

/LF0230/ Ein Benutzer kann sich eine Tafel zur Infografik-Erstellung durch Klick auf diese in der Sammlung auswählen. Es erscheint ein Vorauswahl-Fenster, in dem erste Parametereinstellungen (P1) vorgenommen werden können (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*). [Abb. 7]

3. Infografiken – Vorauswahl

Wurde vom Benutzer eine Thementafel aus seiner „Sammlung“ angeklickt, erhält er eine Voransicht, um erste Parametereinstellungen (P1) vorzunehmen. Diese Parameter werden redaktionell vorbestimmt und stehen individuell für jede Infografik fest (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*).

/LF0310/ Ein Benutzer kann nach abgeschlossener Recherche und Vorauswahl über die „Sammlung“ eine oder nacheinander mehrere Infografiken erstellen und interaktiv damit arbeiten. Dazu klickt er die gewünschte Thementafel an.

/LF0320/ Ein Benutzer erhält über der Zeile der „Sammlung“ durch Klick auf eine Thementafel eine Zusammenstellung des ausgewählten Themas in Form einer rechteckigen Box, in der die zugehörige Miniatur-Infografik, ein kurzer beschreibender Infotext und erste Parameter angezeigt werden.
[Abb. 7]

/LF0330/ Ein Benutzer kann zwischen verschiedenen Diagrammformen auswählen. Es gibt eine Auswahl an 4 verschiedenen Diagrammformen als Icons (Linien, Balken, Kreis, Karte), wobei nur die als Buttons funktionieren, die redaktionell als sinnvoll ernannt wurden (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*). Diese sind dann farbig markiert, die anderen ausgegraut. Bei Klick bemerkt der Benutzer sogleich eine Anpassung in der Miniatur-Grafik.

/LF0340/ Ein Benutzer kann zwischen verschiedenen thematischen Parametern (P1) wählen, die er visualisiert haben möchte (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*). Diese Auswahl geht noch nicht ins absolute Detail, aber macht eine ungefähre Anpassung an persönliche Darstellungswünsche möglich. Bei

Auswahl bemerkt der Benutzer sogleich eine Anpassung in der Miniatur-Grafik.

/LF0350/ Ein Benutzer hat die Möglichkeit, über einen Button erweiterte Einstellungen (P2) vornehmen zu können (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*). [Abb. 8, rechte Seite]

4. Infografiken – Bearbeitung

Hat ein Benutzer in der Voransicht eine Infografik zum weiteren Bearbeiten ausgewählt, vergrößert sich die Infografik, und der Informationstext minimiert sich, bleibt aber aufrufbar [Abb. 8, linke Seite]. Eine Parameter-Toolbox zeigt nun alle möglichen Einstellungsoptionen (P2), aus denen er auswählen kann, um damit die für ihn interessantesten Aspekte herauszuarbeiten [Abb. 8, rechte Seite]. Diese Parameter werden redaktionell vorbestimmt und stehen individuell für jede Infografik fest (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*).

/LF0410/ Ein Benutzer kann in der angezeigten Parameter-Toolbox aus dafür vorgesehenen Optionen auswählen und sieht die Veränderung sofort in der Infografik. Diese Parameter beziehen sich auf die Daten und können z.B. zeitliche, geografische oder bevölkerungswissenschaftliche sein. Die genauen Parameter hängen von der jeweiligen Themen-Auswahl ab und werden zu einem späteren Zeitpunkt detailliert beschrieben (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*).

/LF0420/ Ein Benutzer kann die Parameter-Toolbox minimieren, bzw. einzelne durch Überschriften unterscheidbare Parameter-Kapitel ein- und ausblenden.

/LF0430/ Ein Benutzer kann über die immer als Icon sichtbaren Diagrammtypen zu einem anderen Diagrammtyp wechseln, sofern dieser redaktionell dafür bestimmt wurde (siehe *3) 2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze*).

/LF0440/ Ein Benutzer kann über die immer als Icon sichtbaren Chartformen zu einer anderen Chartform wechseln und damit z.B. eine Multibox erstellen, in der zwei bis vier Infografiken ohne Interaktionsmöglichkeiten neben-/ bzw. übereinander zusammengestellt werden. In diesem Fall muss es möglich sein, jede einzelne Infografik über eine eigene Parameter-Toolbox verändern zu können.

5. Chart Speichern, Chart Merken

Ein Benutzer hat zwei Möglichkeiten, seine bearbeiteten Infografiken abzulegen:

/LF0510/ Ein Benutzer kann über einen Button „Chart merken“ [Abb. 8, linke Seite] seine Infografik mit den aktuell eingestellten Parametern in der „Sammlung“ unter „Infografik“ ablegen. In seiner Sammlung erscheint eine Miniatur-Infografik mit dem dazugehörigen Themenhinweis [Abb. 8, siehe „Sammlung“].

/LF0520/ Ein Benutzer kann über einen Button „Chart speichern“ seine Infografik mit den aktuell eingestellten Parametern in „myZwickr“ unter einem von ihm gewählten Namen speichern. An dieser Stelle wird nach Klicken des „Chart speichern“-Buttons eine Fullscreen-Seite („myZwickr“) angezeigt, die zuvor vollständig angelegt wurde, von daher wird die Speicherung noch nicht automatisch funktionieren.

6. „myZwickr“

myZwickr wird ein simulierter Bereich des Prototypen darstellen, der zuvor fest angelegt wurde. Folgende Funktionen werden ebenfalls angedeutet:

/LF0610/ In „myZwickr“ hat der Benutzer die Möglichkeit, für ihn interessante Infografiken dauerhaft zu speichern.

/LF0620/ In „myZwickr“ hat der Benutzer die Möglichkeit, seine gespeicherten Infografiken herunter zu laden bzw. auszudrucken. [siehe Kap. 3) 1. 7. Download]

/LF0630/ In „myZwickr“ hat der Benutzer die Möglichkeit, für ihn uninteressante Infografiken wieder zu löschen.

7. Download

Um die Infografik in seiner Anwendung weiter verwenden zu können, benötigt der Benutzer die URL einer HTML-Seite, in der das SWF und die Daten (XML) geladen werden. Diese Funktion soll dem Benutzer verdeutlichen, wie einfach eine Weiterverwendung, bzw. Einbettung der generierten Infografik in seine eigene Anwendung funktioniert. Hierzu werden die fünf Datensätze auf einem Server abgelegt, um bei der Präsentation in Form einer URL verwendet werden zu können. Eine Generierung der Infografik mit verwendetem Datensatz ist beim Prototypen nicht vorgesehen.

/LF0710/ Ein Benutzer erhält bei Generierung eine von fünf URLs, um diese in seine Anwendung einbetten zu können.

/LF0720/ Dem Benutzer wird über eine PDF-Grafik angedeutet, dass er seine Infografik als PDF speichern, bzw. drucken kann.

2. Konkrete Benutzerfunktionen am Beispiel ausgesuchter Datensätze

Im Prototypen werden fünf ausgewählte Datensätze als Datengrundlage fungieren, um mögliche Einstellungs- und Bearbeitungsszenarien realistisch demonstrieren zu können. Die einzelnen Infografiken sind beispielhaft in den Anlagen Abb. 9 – Abb. 12 veranschaulicht.

1. Navigationsplan

Die Navigation über 5 ausgewählte funktionierende Pfade wird folgendermaßen umgesetzt:

1. Spätaussiedler (Stat.Bund.Nr.: 12711 – 0010)
   → Gebiet, Bevölkerung, Arbeitsmarkt, Wahlen
   → Bevölkerung
   → Wanderungen
   → Wanderungsstatistik

2. Bundestagswahl (Stat.Bund.Nr.: 14111 – 0001)
   → Gebiet, Bevölkerung, Arbeitsmarkt, Wahlen
   → Wahlen
   → Bundestagswahlen
   → Allgemeine Bundestagswahlstatistik

3. Kindergeld (Stat.Bund.Nr.: 22911 – 0002)
   → Bildung, Sozialleistungen, Gesundheit, Recht
   → Öffentliche Sozialleistungen
   → (Sonstiges im Bereich Sozialleistungen)
   → Statistik über Kindergeld

4. Ankünfte und Übernachtungen in Beherbergungsbetrieben
(Stat.Bund.Nr.: 45412 – 0011)

→ Wirtschaftsbereiche
→ Handel und Instandhaltung, Gastgewerbe, Tourismus
→ Fachstatistiken Handel, Gastgewerbe, Tourismus
→ Monatserhebung im Tourismus

5. Ausgaben für den Umweltschutz
(Stat.Bund.Nr.: 85411 – 0001)

→ Gesamtrechnungen
→ Umweltökonomische Gesamtrechnungen (UGR)
→ Maßnahmen des Umweltschutzes
→ Ausgaben und Anlagevermögen für den Umweltschutz

2. Datensatz 12711-0010, Spätaussiedler

1. Parametereinstellungen (P1)

- Zeit (1997 - 2006):

Standardmäßig wird dem Benutzer beim Aufruf des Datensatzes 12711-0010 ein Liniendiagramm angeboten (Liniendiagramm-Icon ist grau).

- y-Achse Altersgruppen insgesamt
- x-Achse Zeit

/LF1010/ Alternativ kann der Benutzer den gleichen zeitlichen Verlauf in einem Balkendiagramm anzeigen lassen (Balkendiagramm-Icon).

- y-Achse Altersgruppen insgesamt
- x-Achse Zeit

/LF1020/ Der Benutzer kann zwischen Linien- und Balkenansicht hin und her wechseln. Der jeweils angeklickte Button wird in dem Moment inaktiv. Die Miniatur-Ansicht passt sich an die aktuellen Einstellungen an.

- Altersgruppen:
  1. 0 – 6 Jahre
  2. 6 – 18 Jahre
  3. 18 – 25 Jahre
  4. 25 – 45 Jahre
  5. 45 – 65 Jahre
  6. 65+ Jahre

/LF1030/ Der Benutzer kann sich für eine Anzeige der verschiedenen Altersgruppen im Vergleich entscheiden. Das letzte Jahr im Datensatz wird in einem Kuchendiagramm angezeigt.

- 1 Jahr, Kuchenstücke = Anzahl innerhalb der 6 verschiedenen Altersgruppen

2. Erweiterte Parametereinstellungen (P2)

- Zeit:

/LF1040/ Der Benutzer kann nicht nur einzelne Jahre auswählen, sondern über einen Zeitslider den angezeigten Bereich eingrenzen und wieder erweitern. Diese Funktion steht bei Linien- und Balkendiagramm zur Verfügung.

- Zeit/ Altersgruppen:

/LF1050/ Der Benutzer kann bei einem Linien- oder Balkendiagramm über den zeitlichen Parameter hinaus die Grafik nach Altersgruppen sortieren lassen, bzw. Altersgruppen hinzu fügen oder abschalten. Dies bestätigt er mit dem Anklicken einer oder mehrerer Checkboxen. Er erhält ein Liniendiagramm mit einem Verlauf der zeitlichen Entwicklung der ausgewählten Altersgruppe(n).

- Altersgruppen:

/LF1060/ Das ausgewählte Kuchendiagramm zeigt alle Altersgruppen. Über eine Combobox kann der Benutzer je eins von zehn Jahren zur Veränderung des Kuchendiagramms auswählen.

- Diagramme:

/LF1070/ Der Wechsel zwischen Linien- und Balkendiagramm ist permanent möglich.

3. Datensatz 14111-0001, Bundestagswahl Parametereinstellungen

Der Benutzer erhält eine Multibox (vier Quadranten) mit drei Diagrammen, die verschiedene Aspekte des Datensatzes anzeigen. Jedes Diagramm erhält seine eigene Parameter-Toolbox.

- Zeit (2005, 2002, 1998, 1994):

Die Quadranten II und III bilden eine Einheit und zeigen in einem Liniendiagramm die Bundestagswahlergebnisse der letzten 4 Wahljahre (2005, 2002, 1998, 1994) für die Parteien (SPD, CDU/CSU, Grüne, FDP, PDS und Sonstige) im Vergleich.

- y-Achse Parteien
- x-Achse Zeit (Stichtag)

/LF2010/ Über einen Zeitslider kann der Benutzer den angezeigten Bereich eingrenzen und wieder erweitern. Linker und rechter Slider-Kopf dürfen dabei nicht aufeinander gelegt werden können.

/LF2020/ Der Benutzer kann über den zeitlichen Aspekt hinaus über Checkboxen einzelne der 5 Parteien hinzufügen oder abwählen.

/LF2030/ Der Benutzer kann zwischen den Einheiten „gültige Stimmen“ und „Anteil der gültigen Stimmen“ wechseln (Radio-Button).

/LF2040/ Über zwei Diagramm-Icons (Linien/ Balken) kann der Benutzer zwischen Linien- und Balkendiagramm hin und her wechseln.

- Parteien (SPD, CDU/CSU, Grüne, FDP und PDS):

Quadrant I zeigt ein Kuchendiagramm, in dem die Bundestagswahl-ergebnisse der einzelnen Parteien (SPD, CDU/CSU, Grüne, FDP und PDS) zum Zeitpunkt der letzten Wahl (2005) miteinander verglichen werden.

- 1 Jahr, Kuchenstücke = Anteile der Wahlstimmen

/LF2050/ Über eine Combobox kann der Benutzer je eines von vier Wahljahren (Stichtag) zur Veränderung des Kuchendiagramms auswählen.

/LF2060/ Die Kuchenstücke (Parteien) können angeklickt werden und beeinflussen die Deutschlandkarte im IV. Quadranten. Das angewählte Kuchenstück hebt sich optisch vom Rest des Kuchens ab.

- Bundesländer:

Der IV. Quadrant bildet die Karte Deutschlands unterteilt in 16 Bundesländer ab. Rechts daneben befindet sich ein Bereich, in dem tabellarisch Wahlergebnisse und Parteien angezeigt werden können. Zunächst sind alle Bundesländer neutral eingefärbt. Nachdem eine Partei im Kuchen angewählt wurde, zeigt die Karte die Wahlergebnisse der Partei für jedes Bundesland, in dem die Bundesländer unterschiedlich gesättigt eingefärbt werden.

/LF2070/ Bei Rollover über die einzelnen Bundesländer füllt sich die Tabelle mit den Wahlergebnissen des jeweiligen Bundeslandes.

4. Datensatz 22911-0002, Kindergeld

1. Parametereinstellungen (P1)

- Zeit (1965 - 2007):

Standardmäßig werden dem Benutzer beim Aufruf des Datensatzes 22911-0002 zwei mögliche Liniendiagramme angeboten (Liniendiagramm-Icon ist grau).

- y-Achse Werte (2.500 – 30.000)
  - Kinder, für die Kindergeld gezahlt wurde (1000)
  - Ausgezahlte Beiträge (Mio. €)

- x-Achse Zeit

/LF3010/ Alternativ kann sich der Benutzer den gleichen zeitlichen Verlauf in einem Balkendiagramm anzeigen lassen (Balkendiagramm-Icon).

- Kinder/ Beiträge:

/LF3020/ Der Benutzer kann auswählen, dass ihn die Anzahl der Kinder, für die Kindergeld gezahlt wurde, interessiert.

- y-Achse Anzahl der Kinder (1000)
- x-Achse Zeit

/LF3030/ Der Benutzer kann auswählen, dass ihn die ausgezahlten Beiträge interessieren.

- y-Achse Ausgezahlte Beiträge (Mio. €)
- x-Achse Zeit

2. Erweiterte Parametereinstellungen (P2)

- Zeit:

/LF3040/ Der Benutzer kann über eine Combobox einzelne Jahre in einem Balkendiagramm anzeigen lassen.

/LF3050/ Der Benutzer kann über einen Zeitslider den Zeitverlauf eingrenzen und wieder erweitern.

- Kinder/ Beiträge:

/LF3060/ Der Benutzer kann die Anzahl der Kinder, für die Kindergeld gezahlt wurde, und die ausgezahlten Beiträge mit einander vergleichend anzeigen lassen.

- y-Achse
  - Anzahl der Kinder (1000)
  - Ausgezahlte Beiträge (Mio. €)
- x-Achse Zeit (1965 - 2007)

◆ Diagramme:

/LF3070/ Der Wechsel zwischen Linien- und Balkendiagramm ist permanent möglich.

5. Datensatz 45412-0011, Ankünfte und Übernachtungen in Beherbergungsbetrieben

1. Parametereinstellungen (P1)

◆ Zeit (1997 - 2008):

Standardmäßig wird dem Benutzer beim Aufruf des Datensatzes 45412-0011 ein Liniendiagramm angeboten (Liniendiagramm-Icon ist grau).

- y-Achse
  - Ankünfte (Deutschland)
  - Übernachtungen (Deutschland)
- x-Achse
  - 1) Zeit (Jan - Dez 2008)
  - 2) Zeit (1997 - 2008)

/LF4010/ Alternativ kann der Benutzer den gleichen zeitlichen Verlauf in einem Balkendiagramm anzeigen lassen (Balkendiagramm-Icon).

- y-Achse
  - Ankünfte (Deutschland)
  - Übernachtungen (Deutschland)

- x-Achse
  - 1) Zeit (Jan - Dez 2008)
  - 2) Zeit (1997 - 2008)

/LF4020/ Der Benutzer kann zwischen Linien- und Balkenansicht hin und her wechseln. Der jeweils angeklickte Button wird in dem Moment inaktiv. Die Miniatur-Ansicht passt sich an die aktuellen Einstellungen an.

2. Parametereinstellungen (P2)

- Zeit (1):

/LF4030/ Über einen Zeitslider (steht mit beiden Anfassern auf 2008) kann der Benutzer einen beliebig großen Zeitraum zwischen 1997 und 2008 auswählen. Der Verlauf beinhaltet jeweils 12 Monats-Daten pro Jahr.

- Zeit (1/ 2):

/LF4040/ Über 14 Checkboxen kann der Benutzer zu dem gewünschten Zeitbereich auswählen, für welche Monate er die Werte visualisiert haben möchte. 12 Checkboxen stehen für die Monate Januar bis Dezember, 1 Checkbox für alle Monate und 1 Checkbox, falls die Aufteilung in Monate nicht gewünscht wird.

- Herkunft:

/LF4050/ Der Benutzer kann zu den Werten der Ankünfte

und Übernachtungen aller Gäste in Deutschland zusätzlich den Parameter der Herkunft des Gastes hinzufügen. Er erhält zwei zusätzliche Checkboxen, 1 für die Anzahl der Gäste mit Wohnsitz im Ausland und 1 für die Anzahl der Gäste mit Wohnsitz im Inland. Dadurch kann er zwischen 6 Checkboxen wählen:

1) Ankünfte gesamt
2) Übernachtungen gesamt
3) Ankünfte (Gäste mit Wohnsitz im Ausland)
4) Übernachtungen (Gäste mit Wohnsitz im Ausland)
5) Ankünfte (Gäste mit Wohnsitz im Inland)
6) Übernachtungen (Gäste mit Wohnsitz im Inland)

Es ist maximal möglich, vier Parameter gleichzeitig darzustellen (3-6), sobald eine der Gesamt-Auswertungen (1, 2) gewählt ist, sind die anderen Parameter nicht mehr auswählbar.

- Diagramme:

/LF4060/ Der Wechsel zwischen Linien- und Balkendiagramm ist permanent möglich.

6. Datensatz 85411-0001, Ausgaben für den Umweltschutz

1. Parametereinstellungen (P1)

- Zeit (1996 - 2005):

Standardmäßig wird dem Benutzer beim Aufruf des Datensatzes 85411-0001 ein Liniendiagramm angeboten (Liniendiagramm-Icon ist grau).

- y-Achse Werte (2.500 – 37.500)

    - Ausgaben für den Umweltschutz (Mio. €)
    - Investitionen für den Umweltschutz (Mio. €)
    - Laufende Ausgaben für den Umweltschutz (Mio. €)
- x-Achse Zeit

/LF5010/ Alternativ kann der Benutzer den gleichen zeitlichen Verlauf in einem Balkendiagramm anzeigen lassen (Balkendiagramm-Icon).

/LF5020/ Der Benutzer kann zwischen Linien- und Balkenansicht hin und her wechseln. Der jeweils angeklickte Button wird in dem Moment inaktiv. Die Miniatur-Ansicht passt sich an die aktuellen Einstellungen an.

- Anteile:

/LF5030/ Dem Benutzer wird über die Wahl der Darstellung in einem Kuchendiagramm die Möglichkeit geboten, Ausgaben und Investitionen für den Umweltschutz im Vergleich zum BIP zu visualisieren. Dabei wird das standardmäßig das letzte Jahr visualisiert.

2. Parametereinstellungen (P2)

- Zeit:

/LF5040/ Der Benutzer kann über eine Combobox einzelne Jahre in einem Balkendiagramm anzeigen lassen.

/LF5050/ Der Benutzer kann über einen Zeitslider den Zeitverlauf eingrenzen und wieder erweitern.

/LF5060/ Über Checkboxen kann der Benutzer einzelne Anzeige-Parameter ab- oder dazu wählen. Mindestens ein Verlauf muss allerdings angeklickt sein.

- Anteile:

/LF5070/ Der Benutzer kann über eine Combobox einzelne Jahre auswählen, um die Darstellung des Kuchendiagramms zu verändern.

/LF5080/ Der Benutzer kann über eine Combobox zwischen dem Anteil der Ausgaben für den Umweltschutz am BIP und dem Anteil der Investitionen für den Umweltschutz an den Gesamtinvestitionen auswählen. Dies wird für das zuvor gewählte Jahr angezeigt.

- Diagramme:

/LF5090/ Der Wechsel zwischen Linien- und Balkendiagramm ist permanent möglich.

4) Produktdaten

Hinweis: Die einzelnen Produktdaten werden mit einer laufenden Nummer [Bsp.: /LD0100/] versehen, damit alle am Projekt beteiligten Personen alle Daten eindeutig benennen können. „LD“ steht dabei für „Produktdaten des Lastenheftes“.

Da der Prototyp ein User-Interface darstellen soll, sollten folgende Benutzerdaten ohne Funktionalität trotzdem angezeigt werden:

/LD0100/ Benutzerdaten:

- Benutzername, z.B. „Guten Tag Herr Frank Doering“
- Logout-Button

5) Produktleistungen

Hinweis: Die einzelnen Produktleistungen werden mit einer laufenden Nummer [Bsp.: /LL0100/] versehen, damit alle am Projekt beteiligten Personen jede Leistung eindeutig benennen können. „LL“ steht dabei für „Produktleistungen des Lastenheftes“.

An den Prototypen werden folgende Leistungs-Anforderungen gestellt:

/LL0100/ Betriebssystem:
Das Produkt muss plattformübergreifend (Windows, Mac, Linux) einsetzbar sein. Dies ist clientseitig durch die Installation des Adobe Flash Players (mind. Version 9) zu leisten.

6) Qualitätsanforderungen

Der Prototyp soll benutzerfreundlich realisiert werden, sodass der Benutzer durch eine einfache interaktive Funktionalität zur eindeutigen Benutzung der Software unterstützt wird.
Der Prototyp soll effizient funktionieren, daher müssen Navigation und Interaktion ohne größere Wartezeiten reagieren.

7) Entwicklung

Folgende Datei-Formate und Daten werden für die Flex-Entwickler vorbereitet:

1. Daten

Zunächst liegen die Daten der fünf ausgewählten Datensätze als XML-Dateien vor, auf Wunsch können diese formatiert werden.

2. Grafiken

Die Flex-Entwickler arbeiten mit Grafiken auf Grundlagen von Photoshop-Dateien (PSD).

3. Flash

Die Navigationsleiste sowie die „Sammlung“ wird in Adobe Flash erstellt, daraus entstehen funktionstüchtige SWF-Dateien. Somit beginnt der Tätigkeitsbereich der Flex-Entwickler ab dem Klick eines der fünf möglichen Thementafeln in der „Sammlung“.

8) Anlagen

Abb. 1 [Zwickr - Nomenklatur]

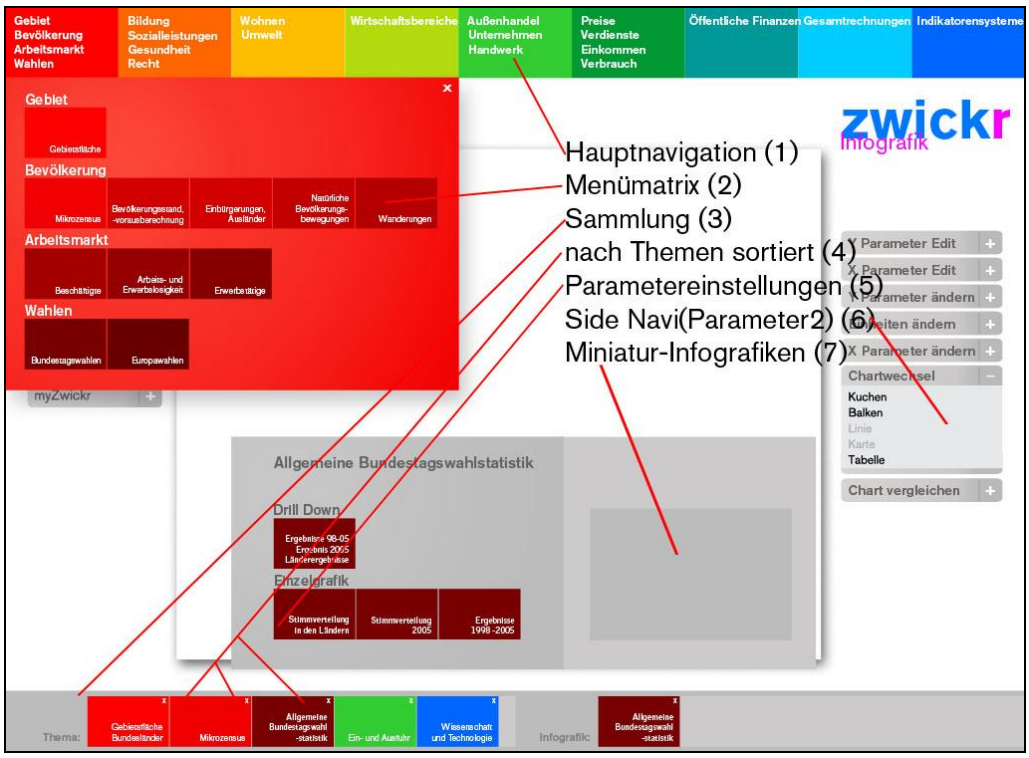


Abb. 2 [Navigation - Hauptmenü]

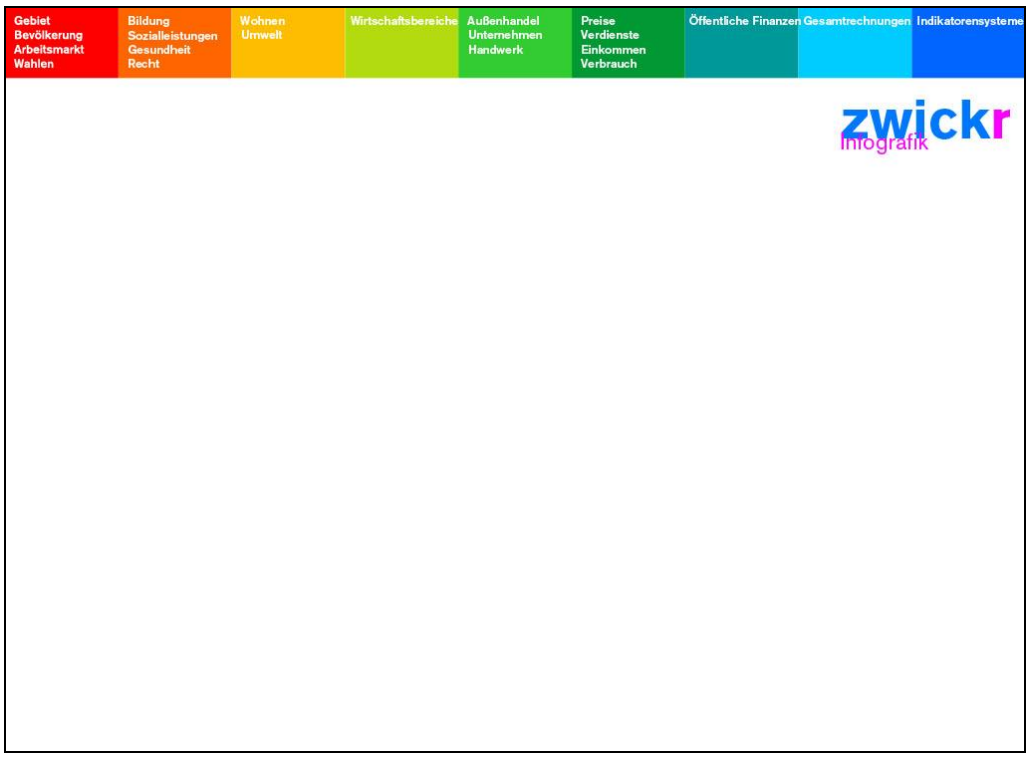


Abb. 3 [Navigation - Menümatrix]

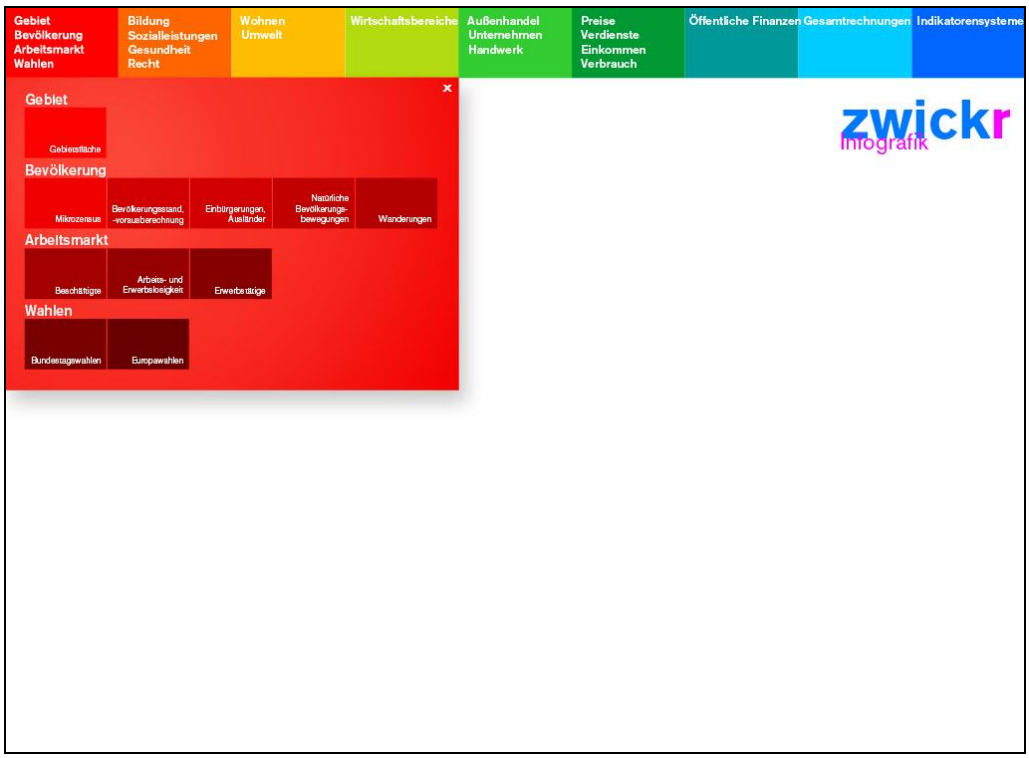


Abb. 4 [Navigation – Themen]

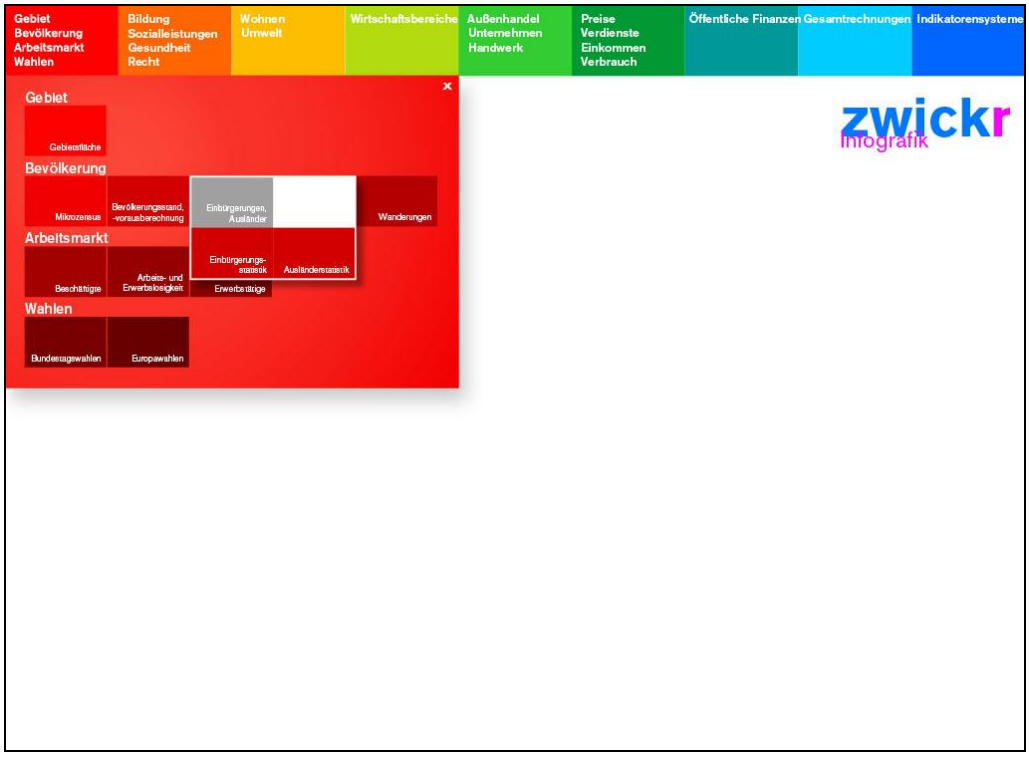


Abb. 5 [Themenauswahl - Sammlung]

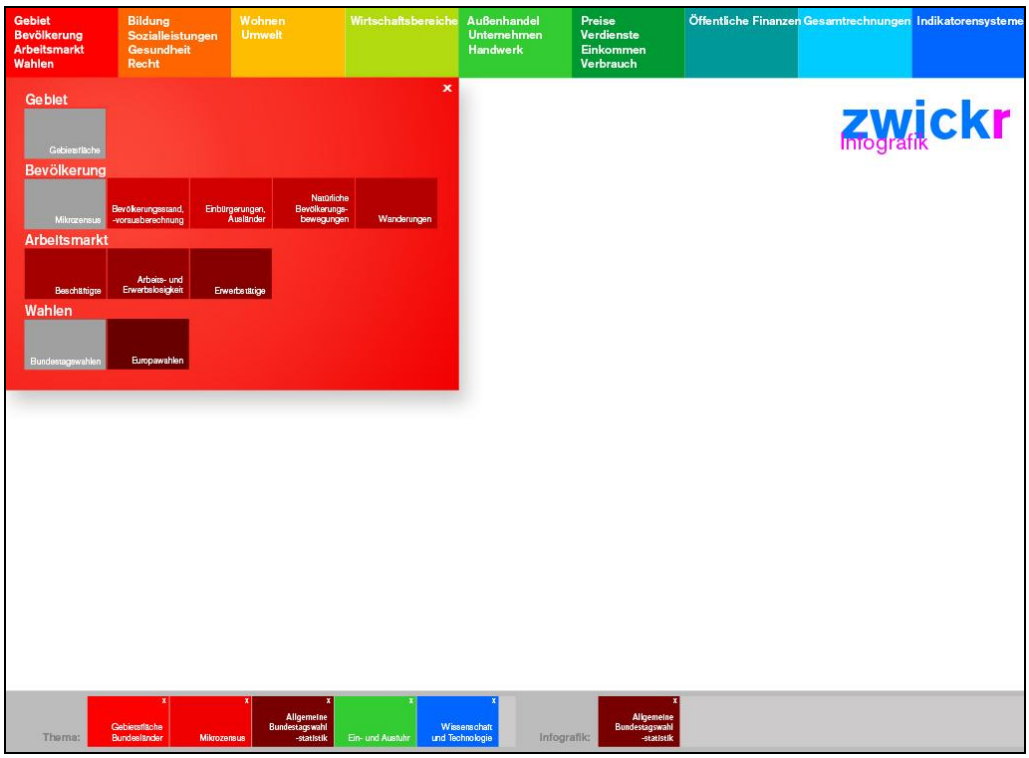


Abb. 6 [Sammlung - Thema/ Infografik]

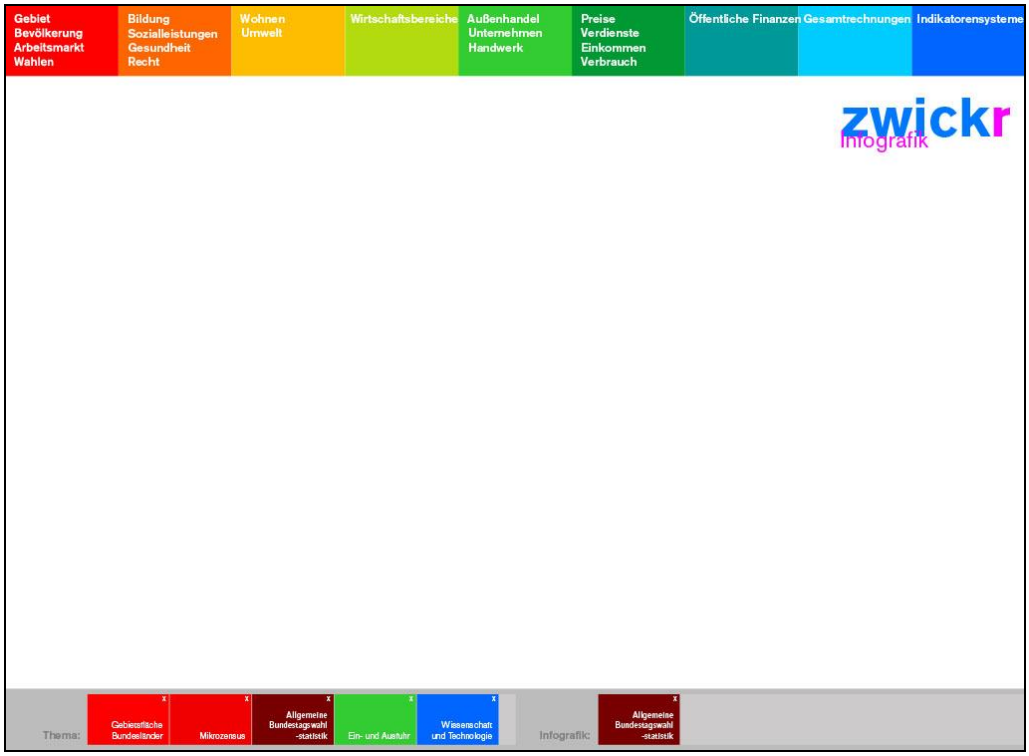


Abb. 7 [Vorauswahl - Parametereinstellungen (P1)]

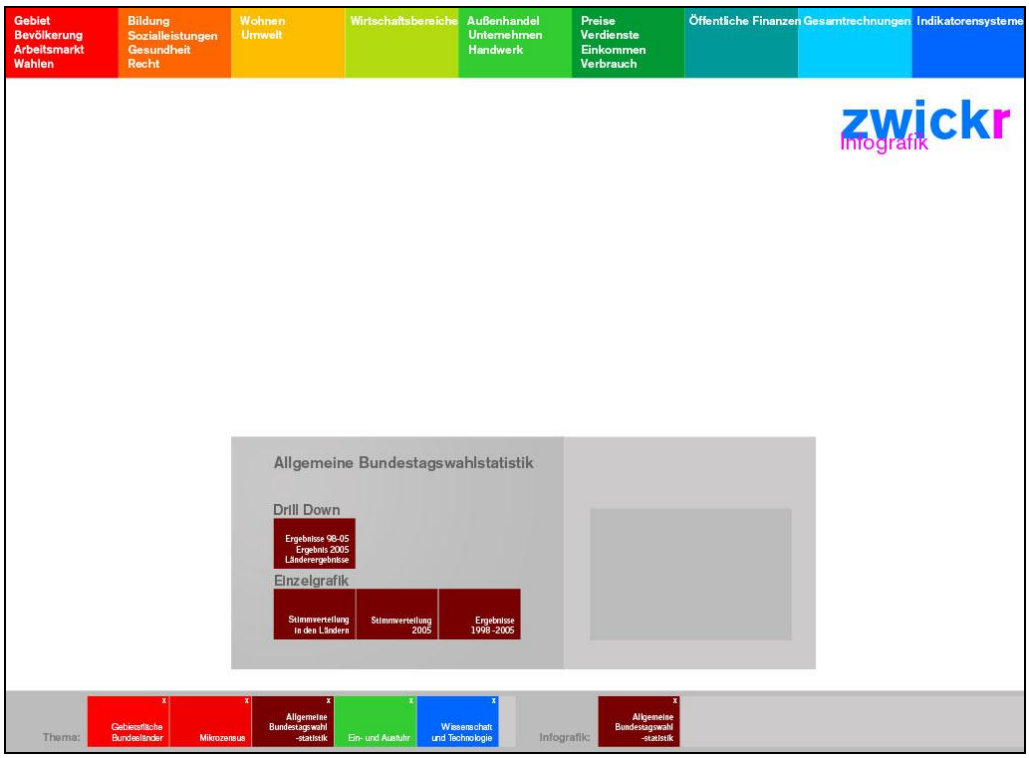


Abb. 8 [Infografikbearbeitung - Parametereinstellungen (P2)]

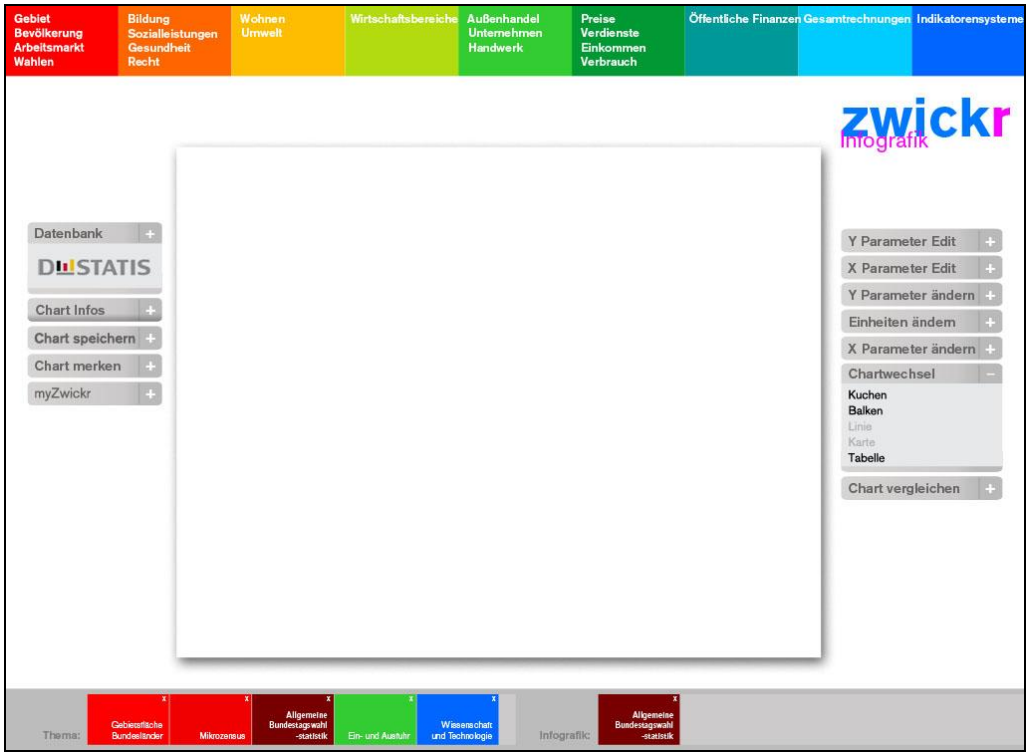


Abb. 9 [Beispiel-Infografik: Bar-Chart]

Abb. 10 [Beispiel-Infografik: Line-Chart]

Abb. 11 [Beispiel-Infografik: Pie-Chart]

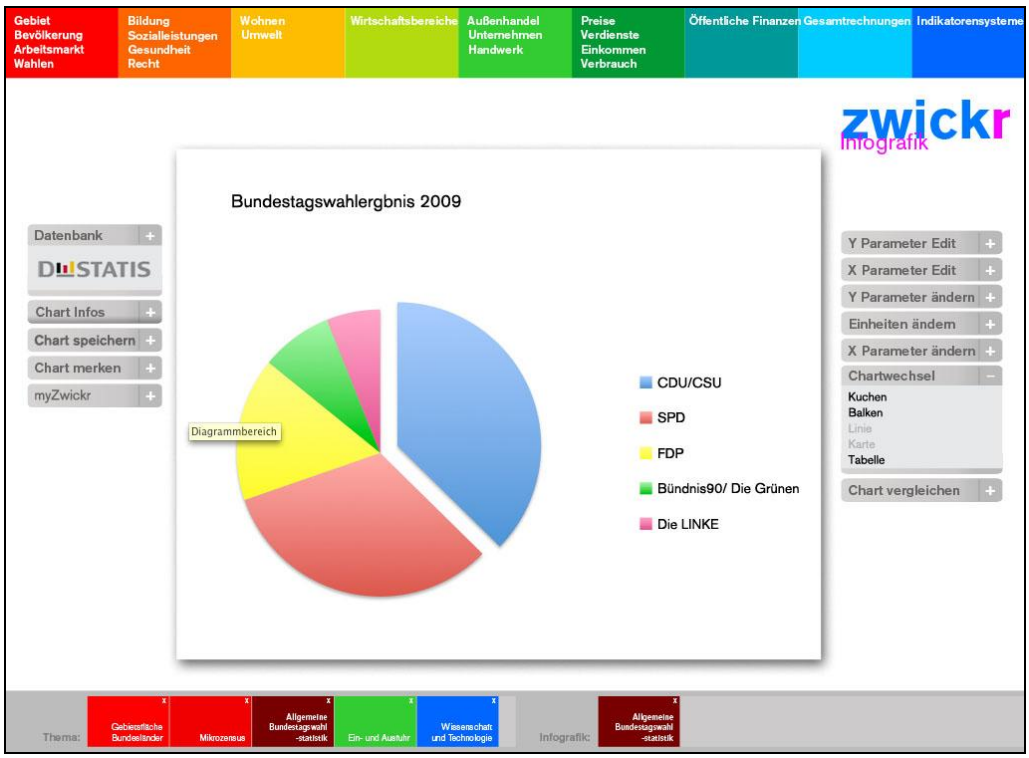
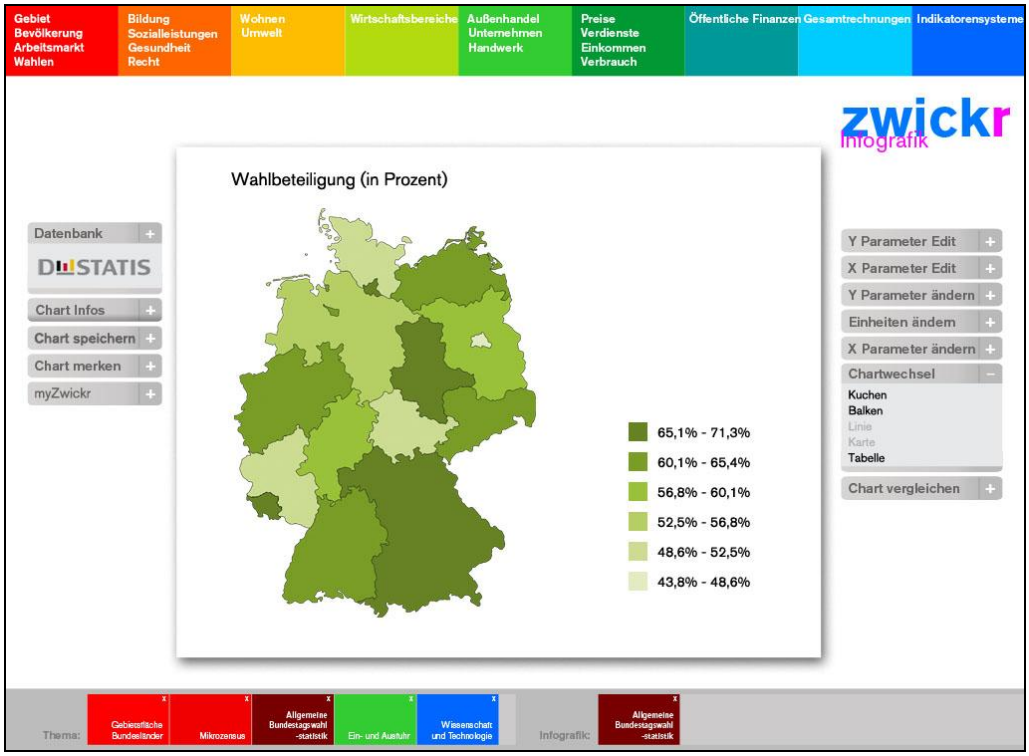


Abb. 12 [Beispiel-Infografik: Map]



Glossar

A

ActionScript

ActionScript (kurz AS) ist eine Programmiersprache des US-amerikanischen Softwareunternehmens Adobe Systems auf Basis des ECMAScript-Standards (ECMA-262) und kann in einer Reihe von Adobe-Produkten eingesetzt werden, so zum Beispiel Adobe Flash, Flex und Air.
→ 43-45, 54, 60

.air

Dateiendung der mit Adobe AIR erstellte Dateien
AIR Applikationen können direkt aus dem Web installiert und gestartet werden. Sie laufen wie herkömmliche Applikationen ohne einen Browser selbstständig auf dem Desktop.
→ 42

AJAX

AJAX ist ein Apronym für die Wortfolge „Asynchronous JavaScript and XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen.
→ 24

Asynchrone Kommunikation

Unter asynchroner Kommunikation versteht man in der Informatik und Netzwerktechnik einen Modus der Kommunikation, bei dem das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch bspw. Warten auf die Antwort des Empfängers (wie es bei synchroner Kommunikation der Fall) stattfindet.
→ 23

B

Best Practice Management

Wenn ein Unternehmen nach best practice vorgeht, setzt es bewährte und kostengünstige Verfahren, technische Systeme und Geschäftsprozesse ein, die es zumindest auf wesentlichen Arbeitsfeldern zum Musterbetrieb für andere machen.
→ 23, 27, 53

Browser

Webbrowser, oder allgemein auch Browser sind spezielle Computerprogramme zum Betrachten von Webseiten im World Wide Web oder allgemein von Dokumenten und Daten.
→ 21, 23, 25

C

Client

Ein Client ist ein Computerprogramm, das Kontakt zu einem anderen Computerprogramm, dem Server, aufnimmt, um dessen Dienstleistung zu nutzen. Ein typisches Beispiel für einen Client ist ein Webbrowser.
→ 23-25, 46

Common Language Runtime

Common Language Runtime (kurz CLR) ist die Laufzeitumgebung von .NET und stellt somit den Interpreter für den standardisierten Zwischencode, der Common Intermediate Language (CIL), dar.
→ 25

Constructive Cost Model

Constructive Cost Model (kurz COCOMO) ist ein algorithmisches Kostenmodell, das in der Softwareentwicklung zur Kosten- bzw. Aufwandsschätzung verwendet wird.
→ 29

Character-Separated Values

Das Dateiformat Character-Separated Values (kurz CSV) beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.
→ 17

Compiler

Ein Compiler, auch Übersetzer oder Kompilierer genannt, ist ein Computerprogramm, das ein in einer Quellsprache geschriebenes Programm – genannt Quellprogramm – in ein semantisch äquivalentes Programm einer Zielsprache (Zielprogramm) umwandelt.
→ 42, 45, 48,

Cascading Style Sheets

Cascading Style Sheets (kurz CSS) ist eine deklarative Stylesheet-Sprache für strukturierte Dokumente. Sie wird vor allem zusammen mit HTML und XML eingesetzt.
→ 42,

D Datenkapselung

Als Datenkapselung bezeichnet man in der Programmierung das Verbergen von Daten oder Informationen vor dem Zugriff von außen. Der direkte Zugriff auf die interne Datenstruktur wird unterbunden und erfolgt stattdessen über definierte Schnittstellen (Black-Box-Modell).
→ 35

Debugger

Ein Debugger ist ein Werkzeug zum Diagnostizieren, Auffinden und Beheben von Fehlern in Computersystemen, dabei vor allem in Programmen.
→ 42

Design Pattern

Entwurfsmuster (engl. design pattern) sind bewährte Lösungs-Schablonen für wiederkehrende Entwurfsprobleme in Softwarearchitektur und Softwareentwicklung.
→ 37, 60

E Eclipse

Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art.
→ 42

ECMAScript

Der als ECMAScript (ECMA 262) standardisierte Sprachkern von JavaScript beschreibt eine moderne, schlanke, dynamisch typisierte, objektorientierte, aber klassenlose Skriptsprache, die dennoch allen objektorientierten Programmierparadigmen unter anderem auch – aber eben nicht ausschließlich – auf der Basis von Prototypen gerecht wird.
→ 44

Event-Driven Architecture

Die Event-Driven Architecture (kurz EDA) oder auch Ereignisgetriebene Architektur ist ein Softwarearchitektur-Muster, in dem das Zusammenspiel der Komponenten durch Ereignisse (Events) gesteuert wird.
→ 35

Excel

Excel ist ein Tabellenkalkulationsprogramm von Microsoft. Es ist heute die meistverbreitete Software für Tabellenkalkulation.
→ 19

Extreme Programming

Extreme Programming (XP), auch Extremprogrammierung, ist eine Methode, die das Lösen einer Programmieraufgabe in den Vordergrund der Softwareentwicklung stellt und dabei einem formalisierten Vorgehen geringere Bedeutung zumisst.
→ 31-33

F

Flash

Flash von Adobe Systems ist eine proprietäre integrierte Entwicklungsumgebung zur Erstellung multimedialer Inhalte, der Flash-Filme.
→ 24-26, 42-43, 45, 54, 61, 85

Flash Player

Flash Player von Adobe Systems ist ein Abspielprogramm für Flash-Filme, das auch als Plug-in in den Webbrowser eingebunden werden kann.
→ 26, 42, 45

Flex

Flex ist ein Entwicklungsframework von Adobe Systems zum Erstellen von Rich Internet Applications (RIAs). Das Framework besteht aus dem Software Development Kit, dem Flex Builder, dem LiveCycle Data Service und den Flex Charting Komponenten.
→ 24-26, 42-43, 45-46, 48-49, 51, 54, 61-62, 67, 84-85

Framework

Ein Framework (engl. für „Rahmenstruktur, Fachwerk“) ist ein Programmiergerüst, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird.
→ 16, 21, 23-26, 42, 49, 51, 85

G

Gantt-Diagramm

Ein Gantt-Diagramm oder Balkenplan ist ein nach dem Unternehmensberater Henry L. Gantt (1861–1919) benanntes Instrument des Projektmanagements, das die zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellt.
→ 40

H

Hypertext Markup Language

Die Hypertext Markup Language (kurz HTML) ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten.
→ 11, 19, 23, 34, 51

Hypertext Transfer Protocol

Das Hypertext Transfer Protocol (kurz HTTP) ist ein Protokoll zur Übertragung von Daten über ein Netzwerk. Es wird hauptsächlich eingesetzt, um Webseiten aus dem World Wide Web in einen Webbrowser zu laden.
→ 25, 46-47

I Integrated Development Environment

Eine integrierte Entwicklungsumgebung Integrated Development Environment
Eine IDE, eine integrierte Entwicklungsumgebung, ist ein Anwendungsprogramm zur Entwicklung von Software.
→ 25

Interface

Die Schnittstelle oder das Interface ist der Teil eines Systems, der der Kommunikation dient. Softwareschnittstellen oder softwareseitige Datenschnittstellen sind logische Berührungspunkte in einem Softwaresystem: sie definieren, wie Kommandos und Daten zwischen verschiedenen Prozessen und Komponenten ausgetauscht werden.
→ 21

J Java

Java ist eine objektorientierte Programmiersprache und als solche ein eingetragenes Warenzeichen der Firma Sun Microsystems. Sie ist eine Komponente der Java-Technologie.
→ 24, 41,

JavaScript

JavaScript ist eine Skriptsprache, die hauptsächlich für das DOM-Scripting in Web-Browsern eingesetzt wird. Dabei ist unter JavaScript die Gesamtheit aus den Eigenschaften des Browsers sowie des Document Object Models (DOM) und des Sprachkerns zu verstehen.
→ 24, 44

L Lastenheft

Ein Lastenheft (teils auch Anforderungsspezifikation, Kundenspezifikation oder Requirements Specification) beschreibt die Gesamtheit der Forderungen des Auftraggebers an die Lieferungen und Leistungen eines Auftragnehmers.
→ 29, 39, 41, 65

Lose gekoppelte Programmierung

Lose Kopplung bezeichnet in der Informatik einen geringen Grad der Abhängigkeit mehrerer Hard- oder Software-Komponenten untereinander. Bei loser Kopplung eines Systems lassen sich Änderungen einzelner Komponenten oftmals einfacher durchführen, da die Änderung nur eine lokale Auswirkung hat.
→ 23, 35, 49

M Metadaten

Als Metadaten oder Metainformationen bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten.
→ 18-20

Milestone

Ein Milestone, übersetzt Meilenstein, ist ein Ereignis besonderer Bedeutung. Im Projektmanagement sind diese Ereignisse meist Unter- bzw. Zwischenziele eines Projektes.
→ 5, 30,

MVC

Abkürzung für *Model View Controler*
MVC ist ein Architekturmuster zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell (engl. model), Präsentation (engl. view) und Programmsteuerung (engl. controller).
→ 37-38

MXML

MXML ist eine XML-basierte, deklarative (beschreibende) Programmiersprache. Ähnlich wie in HTML werden mit MXML sichtbare und unsichtbare Komponenten beschrieben.
→ 25, 43, 45, 47

N .NET

.NET ist eine von Microsoft entwickelte Software-Plattform.
→ 25, 46

O Objektorientierte Programmierung

Die objektorientierte Programmierung (OOP) ist ein auf dem Konzept der Objektorientierung basierendes Programmierparadigma. Die Grundidee der objektorientierten Programmierung ist, Daten und Funktionen, die auf diese Daten angewandt werden können, möglichst eng in einem sogenannten Objekt zusammenzufassen und nach außen hin zu kapseln, so dass Methoden fremder Objekte diese Daten nicht versehentlich manipulieren können.
→ 23, 34-35

P PHP

PHP ist ein rekursives Akronym für „Hypertext Preprocessor“ und ein Backronym aus „Personal Home Page Tools“
Es ist eine Skriptsprache mit einer an C angelehnten Syntax, die hauptsächlich zur Erstellung von dynamischen Webseiten oder Webanwendungen verwendet wird. PHP ist Open-Source-Software.
→ 25, 46, 84

Plug-in

Ein Plug-in ist ein Computerprogramm, das in ein anderes Softwareprodukt „eingeklinkt“ wird und damit dessen Funktionalität erweitert.
→ 25-26

Polymorphie

Polymorphie (griechisch, „Vielgestaltigkeit“) ist ein Konzept von Programmiersprachen. Es beschreibt die Fähigkeit eines Bezeichners – der einen Festwert (Literal) oder eine Variable repräsentieren kann – sich abhängig von seiner Verwendung unterschiedlich darzustellen. Sie erlaubt dem Bezeichner, je nach Kontext einen unterschiedlichen Datentypen anzunehmen.
→ 35

Prototyping

Prototyping bzw. Prototypenbau ist eine Methode der Softwareentwicklung, die schnell zu ersten Ergebnissen führt und frühzeitiges Feedback bezüglich der Eignung eines Lösungsansatzes ermöglicht.
→ 32, 51, 64, 84

R **REST**

REST ist ein Akronym für die Wortfolge „Representational State Transfer“ und bezeichnet einen Softwarearchitekturstil für verteilte Hypermedia-Informationssysteme wie das World Wide Web. Während dessen Architektur durch den URI-Standard und HTTP beschrieben werden kann, legt der REST-Architekturstil nahe, jede Ressource mit einem eigenen URI anzusprechen.
→ 24

Rich Internet Application

Der Begriff Rich Internet Application (RIA, deutsch: reichhaltige Internet-Anwendung) beschreibt eine Anwendung, die Internet-Techniken benutzt und eine intuitive Benutzeroberfläche bietet.
→ 2, 16, 22-26, 34, 42, 84

S **SCM**

Abkürzung für *Software Configuration Management*
Das SCM oder Softwarekonfigurationsmanagement ist eine Spezialisierung des Konfigurationsmanagements auf alle Aktivitäten im Bereich der Software-Entwicklung.
→ 28

Server

Ein Server (Software) ist ein Programm, das mit einem anderen Programm, dem Client, kommuniziert, um ihm Zugang zu speziellen Dienstleistungen (genannt Dienste) zu verschaffen.
→ 23-25, 46

Silverlight

Silverlight von Microsoft ist ein proprietäres, programmierbares Browser-Plug-in unter Windows und Apple Macintosh. Zweck von Silverlight ist die Ausführung von Rich Internet Applications, die auf Basis der .NET-Plattform geschrieben wurden.
→ 24-26, 84

SOAP

SOAP, „Simple Object Access Protocol“, ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf folgende Standards: XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht zur Übertragung der Nachrichten.
→ 24, 46, 48

.swf

Die aus Flash resultierenden Dateien liegen im SWF-Format vor, einem auf Vektorgrafiken basierenden Grafik- und Animationsformat. Das Kürzel SWF steht dabei für Shockwave Flash.
→ 42, 45, 61

U UI

Abkürzung für *User Interface*
Mit UI, bzw. Benutzerschnittstelle wird das Untersystem in einem Mensch-Maschine-System bezeichnet, mit dem Menschen interagieren.
→ 26

UML

Abkürzung für *Unified Modeling Language*
Die UML ist eine von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software und anderen Systemen.
→ 35

Unittest

Ein Modultest, auch als Unittest bezeichnet) ist Teil des Softwaretests. Er dient zur Verifikation der Korrektheit von Modulen einer Software, z. B. von einzelnen Klassen. Nach jeder Änderung sollte durch Ablauf aller Testfälle nach Programmfehlern gesucht werden.
→ 33

V Version Control System

Ein Version Control System (VCS) ist ein System, das zur Versionierung und Aktualisierung von Quelltexten verwendet wird.
→ 29

W WPF/E

Abkürzung für *Windows Presentation Foundation/ Everywhere*
WPF/E war der ursprüngliche Name von Microsoft Silverlight.
→ 24

X XAML

Abkürzung für *Extensible Application Markup Language*
XAML wurde von Microsoft entwickelt und ist eine neue allgemeine Beschreibungssprache für die Oberflächengestaltung von Anwendungen.
→ 25

XML

Abkürzung für *Extensible Markup Language*
XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten. XML wird u. a. für den Austausch von Daten zwischen Computersystemen eingesetzt, speziell über das Internet.
→ 24-25, 43, 46-47

Abkürzungen

A **AS**
→ ActionScript

AIR
→ Adobe Integrated Runtime

AJAX
→ Asynchronous JavaScript and XML

AMF
→ Action Message Format

C **CLR**
→ Common Language Runtime

COCOMO
→ Constructive Cost Model

CSS
→ Cascading Style Sheets

CSV
→ Character-Separated Values

E **ECMA**
→ European Computer Manufacturers Association

EDA
→ Event-Driven Architecture

G **GIS**
→ Geoinformationssystem

H **HTML**
→ Hypertext Markup Language

HTTP
→ Hypertext Transfer Protocol

HTTPS
→ Hypertext Transfer Protocol Secure

I **ID**

→ Identifier

IDE

→ Integrated Development Environment

M **MVC**

→ Model View Controller

O **OOP**

→ Objektorientierte Programmierung

P **PHP**

→ Personal Home Page Tools

R **REST**

→ Representational State Transfer

RIA

→ Rich Internet Application

RPC

→ Remote Procedure Call

S **SCM**

→ Software Configuration Management

SOAP

→ Simple Object Access Protocol

U **UI**

→ User Interface

UML

→ Unified Modeling Language

URI

→ Universal Resource Identifier

URL

→ Uniform Resource Locator

V **VCS**

→ Version Control System

W **W3C**

→ World Wide Web Consortium

WPF/E

→ Windows Presentation Foundation/ Everywhere

X **XAML**

→ Extensible Application Markup Language

XML

→ Extensible Markup Language

XP

→ Extreme Programming

Abbildungen

Tabellen

Quellcode

Literatur

[AOS09] Adobe Online Store – Deutschland

https://store2.adobe.com/cfusion/store/index.cfm?store=OLS-DE&view=ols_prod&category=/Applications/FlexBuilderStandard&distributionMethod=FULL&nr=0#view=ols_prod&category=/Applications/FlexBuilderStandard&store=OLS-DE&loc=de_de
[Stand: 30.07.2009]

[B09] Baur, Stefan
Lastenheft

http://www.stefan-baur.de/cs.se.lastenheft.html?glstyle=2008
[Stand: 06.02.2009]

[CoD08] Create or Die, Software & Support Verlag GmbH
Harry Klein:
Rich Internet Applications – Ein Überblick

http://createordie.de/cod/artikel/Rich-Internet-Applications-%26ndash%3B-Ein-Ueberblick-1791.html
[Juni 2008]

[Dae09] Dässler, Rolf (FH Potsdam):
Informationsvisualisierung – Stand, Kritik und Perspektiven

http://fabdp.fh-potsdam.de/infoviz/paper/InfoVis99.pdf
[Stand: 29.06.2009]

[DD1780] D'Alembert, Jean Baptiste le Rond/ Diderot, Denis:
Encyclopédie, ou dictionnaire raisonné des sciences, des arts et des métiers, 35 Bde, 1751 – 1780
Harald Fischer Verlag, 1996

[F04] Dr. Flick, Claudia
GENESIS-Online
Die Internetdatenbank des Statistischen Bundesamtes

http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Content/Publikationen/Querschnittsveroeffentlichungen/WirtschaftStatistik/Datenbank/Genesisonline,property=file.pdf
[2004]

[FD06] Friendly, Michael/ Denis, Daniel J.:
Milestones in the history of thematic cartography, statistical graphics, and data visualization
http://curvebank.calstatela.edu/greatlinks/milestone.pdf
[July 5, 2006]

[Ho08] Heise online - iX
Flex vs. Silverlight: Unterschiede und Gemeinsamkeiten
http://www.heise.de/ix/artikel/2008/08/042/
[August 2008]

[LR09] Lahres, Berhard/ Rayman Gregor
Galileo <openbook>
Objektorientierte Programmierung
http://openbook.galileocomputing.de/oop/
[Stand: 30.07.2009]

[MLo09a] Mayers Lexikon online: *Zeichen*.
http://lexikon.meyers.de/wissen/Zeichen+(Sachartikel)+Informatik
[Stand: 06.02.2009]

[MLo09b] Mayers Lexikon online: *Daten*.
http://lexikon.meyers.de/wissen/Daten+(Sachartikel)+Informatik
[Stand: 06.02.2009]

[MLo09c] Mayers Lexikon online: *Information*.
http://lexikon.meyers.de/wissen/Information+(Sachartikel)+Informatik_Elektrotechnik
[Stand: 06.02.2009]

[P05] Playfair, William:
Commercial and Political Atlas and Statistical Breviary
Cambridge University Press, 2005

[RS04] Rembold, Magnus/ Späth, Jürgen:
Munterbund
http://www.munterbund.de/visualisierung_textaehnlichkeiten/artikel.php
[2004]

[S09] Schneeberger, Arnold:
Best Practice Management
http://ssw.jku.at/Teaching/Lectures/Sem/2002/reports/Schneeberger/index.html
[Stand: 18.07.09]

[VSS07] Völker, Rainer/ Sauer, Siegrid/ Simon, Monika:
Wissensmanagement im Innovationsprozess (Kap. 3.1.1)
Physica-Verlag, 2007

[W08a] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 15.2)
Carl HanserVerlag, 2008

[W08b] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 1.2)
Carl HanserVerlag, 2008

[W08c] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 13.1)
Carl HanserVerlag, 2008

[W08d] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 13.2.1)
Carl HanserVerlag, 2008

[W08e] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 13.2.2)
Carl HanserVerlag, 2008

[W08f] Widjaja, Simon:
Rich Internet Applications mit Adobe Flex 3 (Kap. 13.2.3)
Carl HanserVerlag, 2008

[W09a] Wikipedia
AJAX
`http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29`
[Stand: 30.07.2009]

[W09b] Wikipedia
Best Practice
`http://de.wikipedia.org/wiki/Best_Practice`
[Stand: 30.07.2009]

[W09c] Wikipedia
Extreme Programming
`http://de.wikipedia.org/wiki/Extreme_Programming`
[Stand: 30.07.2009]

[W09d] Wikipedia
Objekorientierte Programmierung
http://de.wikipedia.org/wiki/Objektorientierte_Programmierung
[Stand: 30.07.2009]

[W09e] Wikipedia
Lose Kopplung
http://de.wikipedia.org/wiki/Lose_Kopplung
[Stand: 30.07.2009]

[W09f] Wikipedia
Model View Controler
http://de.wikipedia.org/wiki/MVC
[Stand: 30.07.2009]

[W09g] Wikipedia
Gantt-Diagramm
http://de.wikipedia.org/wiki/Gantt-Diagramm
[Stand: 30.07.2009]

Eidesstattliche Erklärung

Köln, 28. August 2009

Ich, Janina Alexandra Maria Werner (Studentin der Medientechnik an der Fachhochschule Köln, Matr.Nr. 11048027), versichere hiermit eidesstattlich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Janina Werner

Sperrvermerk

Die Einsicht in die vorgelegte Arbeit ist bis zum 28.08.2011 gesperrt.